软件开发视频大讲堂

本丛书第1版4种被评为
全国优秀畅销书

PHP 从入门到精通

潘凯华 刘中华 等编著

38 小时语音视频讲解

☑模块库 ☑案例库 ☑题库 ☑素材库

（第2版）



DVD语音视频教学光盘

38小时教学视频录像，全程语音讲解
本书实例源程序、相关素材

本书特色

基础知识→核心技术→高级应用→项目实战
309个应用实例，41个典型应用，4个项目案例
内容极为详尽，实例典型丰富

全程技术服务

答疑网站：www.mingribook.com
提供模块库、案例库、题库、素材库、答疑服务

清华大学出版社

软件开发视频大讲堂

PHP 从入门到精通

（第 2 版）

潘凯华　刘中华　等编著

清华大学出版社

北　京

内 容 简 介

本书从初学者角度出发，通过通俗易懂的语言，丰富多彩的实例，详细介绍了使用 PHP 进行网络开发应该掌握的各方面技术。全书共分 24 章，包括初识 PHP、PHP 环境搭建和开发工具、PHP 语言基础、流程控制语句、字符串操作、正则表达式、PHP 数组、PHP 与 Web 页面交互、PHP 与 JavaScript 交互、日期和时间、Cookie 与 Session、图形图像处理技术、文件系统、面向对象、PHP 加密技术、MySQL 数据库基础、phpMyAdmin 图形化管理工具、PHP 操作 MySQL 数据库、ADODB 类库、Zend Framework 框架、Smarty 模板技术、PHP 与 XML 技术、PHP 与 Ajax 技术、应用 Smarty 模板开发电子商务网站等。书中所有知识都结合具体实例进行介绍，涉及的程序代码均附以详细的注释，可以使读者轻松领会 PHP 程序开发的精髓，快速提高开发技能。

本书列举了大量的小型实例、综合实例和部分项目案例；所附 DVD 光盘内容有同步视频讲解、实例源程序、“实践与练习”答案等；本书的服务网站提供了模块库、案例库、题库、素材库、答疑服务。

本书内容详尽，实例丰富，非常适合作为编程初学者的学习用书，也适合作为开发人员的查阅、参考资料。

图书在版编目（CIP）数据

PHP 从入门到精通/潘凯华，刘中华等编著．—2 版．—北京：清华大学出版社，2010.7
（软件开发视频大讲堂）

ISBN 978-7-302-22747-2

I. ①P…　II. ①潘…　②刘…　III. ①PHP 语言-程序设计　IV. ①TP312

中国版本图书馆 CIP 数据核字（2010）第 081734 号

责任编辑：刘利民　朱　俊
版式设计：牛瑞瑞
责任校对：柴　燕
责任印制：
出版发行：清华大学出版社　　**地　　址：**北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：203×260　**印　　张：**36　**字　　数：**961 千字
（附 DVD 视频光盘 1 张）
版　　次：2010 年 7 月第 2 版　**印　　次：**2010 年 7 月第 1 次印刷
印　　数：1～5000
定　　价：69.80 元

产品编号：035720-01

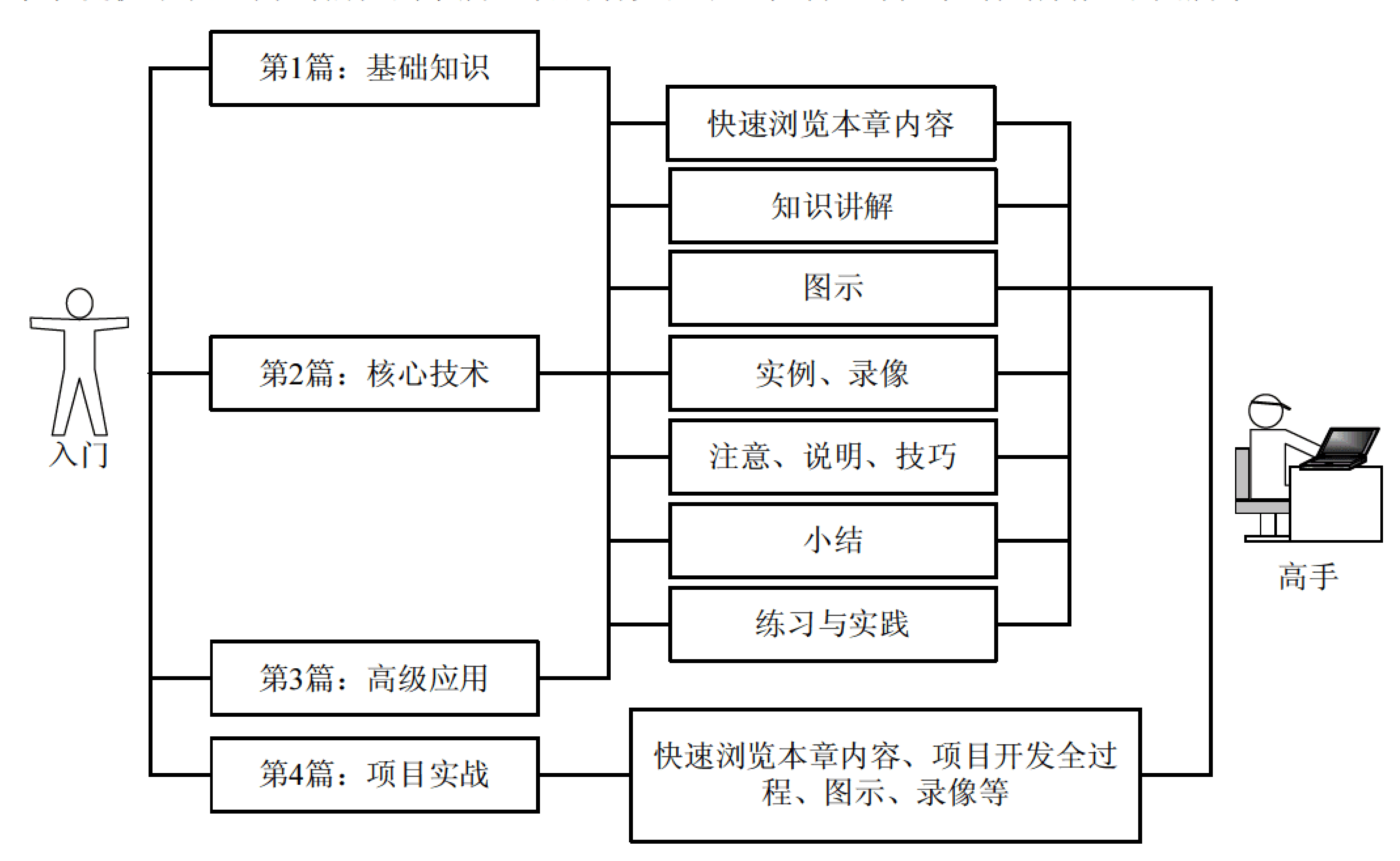
前　言

Preface

PHP 是全球最普及、应用最广泛的互联网开发语言之一。PHP 语言具有简单、易学、源码开放、可操纵多种主流与非主流的数据库、支持面向对象的编程、支持跨平台的操作以及完全免费等特点，越来越受到广大程序员的青睐和认同。PHP 目前拥有几百万名用户，其发展速度要快于它之前的任何一种计算机语言，相信 PHP 在经过不断发展后，一定会成为互联网开发语言中“主流的主流”。

本书内容

本书提供了从入门到编程高手所必备的各类知识，共分 4 篇，大体结构如下图所示。



第 1 篇：基础知识。本篇通过初识 PHP、PHP 环境搭建和开发工具、PHP 语言基础、流程控制语句、字符串操作、正则表达式、PHP 数组、PHP 与 Web 页面交互、PHP 与 JavaScript 交互、日期和时间等章节，并结合大量的图示、举例、录像等，使读者快速掌握 PHP 语言，并为以后编程奠定坚实的基础。

第 2 篇：核心技术。本篇介绍了 Cookie 与 Session、图形图像处理技术、文件系统、面向对象、PHP 加密技术、MySQL 数据库基础、phpMyAdmin 图形化管理工具、PHP 操作 MySQL 数据库、ADODB 类库、Zend Framework 框架等。学习完本篇内容，能够开发数据库应用程序和一些中小型的热点模块。

第 3 篇：高级应用。本篇介绍了 Smarty 模板技术、PHP 与 XML 技术和 PHP 与 Ajax 技术等。学习

完本篇内容，能够开发一些实用的网络程序等。

第 4 篇：项目实战。本篇通过 Smarty 模板技术、ADODB 类库、Ajax 等主流技术实现一个大型、完整的电子商务平台，运用软件工程的设计思想，让读者学习如何进行网站项目的实践开发。书中按照编写项目计划书→系统设计→数据库设计→创建项目→实现项目→运行项目→解决开发常见问题→发布网站的过程进行介绍，带领读者一步步亲身体验开发项目的全过程。

本书特点

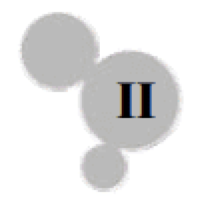
- **由浅入深，循序渐进：**本书以初中级程序员为对象，先从 PHP 基础学起，再学习 PHP 的核心技术，然后学习 PHP 的高级应用，最后学习开发一个完整项目。讲解过程中步骤详尽，版式新颖，在操作的内容图片上以“❶❷❸……”编号+内容的方式进行标注，让读者在阅读时一目了然，从而快速掌握书中内容。
- **语音视频，讲解详尽：**书中每一章节均提供声图并茂的语音视频教学录像，读者可以根据书中提供的录像位置在光盘中找到。这些录像能够引导初学者快速入门，感受编程的快乐和成就感，增强进一步学习的信心，从而快速成为编程高手。
- **实例典型，轻松易学：**通过例子学习是最好的学习方式，本书通过“一个知识点、一个例子、一个结果、一段评析、一个综合应用”的模式，透彻详尽地讲述了实际开发中所需的各类知识。另外，为了便于读者阅读程序代码，快速学习编程技能，书中几乎每行代码都提供了注释。
- **精彩栏目，贴心提醒：**本书根据需要在各章安排了很多“注意”、“说明”、“技巧”等小栏目，以让读者在学习过程中更轻松地理解相关知识点及概念，更快地掌握个别技术的应用技巧。
- **应用实践，随时练习：**书中几乎每章都提供了“练习与实践”，让读者能够通过对问题的解答重新回顾、熟悉所学知识，举一反三，为进一步学习做好充分的准备。

读者对象

- ☑ 初学编程的自学者
- ☑ 编程爱好者
- ☑ 大中专院校的老师和学生
- ☑ 相关培训机构的老师和学员
- ☑ 毕业设计的学生
- ☑ 初中级程序开发人员
- ☑ 程序测试及维护人员
- ☑ 参加实习的“菜鸟”程序员

读者服务

为了方便读者，本书提供了学习答疑网站：www.mingribook.com。有关本书内容的问题读者均可在网站上留言，我们力求在 24 小时内回复（节假日除外）。



致读者

本书由 PHP 网络开发团队组织编写，主要编写人员有潘凯华、刘中华、邹天思、孙鹏、顾彦玲、杨丽、刘欣、刘玲玲、梁晓岚、黄锐、孙明娇、寇长梅、张鹏斌、董大永、吕继迪、张艳、郭佳博、乔敏、梁水、陈丹丹、吕双、张仿彦、徐薇、陈紫宏、唐政、房大伟、张领、苗春义、马文强、王殊宇、李言、李贺、刘锐宁、周桓、张金辉、王永生、王茜等。在编写本书的过程中，我们以科学、严谨的态度，力求精益求精，但错误、疏漏之处在所难免，敬请广大读者批评指正。我们的服务邮箱是 tmoonbook@sina.com、th_press@263.net，读者在阅读本书时，如果发现错误或遇到问题，可以发送电子邮件及时与我们联系，我们会尽快给予答复。

感谢您购买本书，希望本书能成为您编程路上的领航者。

“零门槛”编程，一切皆有可能。

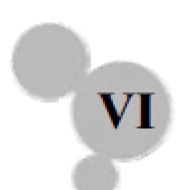
祝读书快乐！

编　者

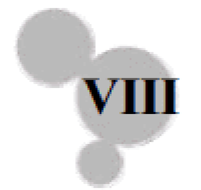
目　录

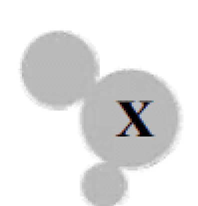
Contents

第 1 篇　基础知识



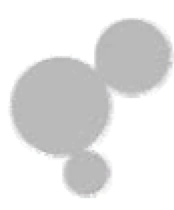
第 2 篇 核心技术





第 3 篇　高级应用

第 4 篇　项目实战



第 1 篇

基础知识

本篇通过对初识 PHP、PHP 环境搭建和开发工具、PHP 语言基础、流程控制语句、字符串操作、正则表达式、PHP 数组、PHP 与 Web 页面交互、PHP 与 JavaScript 交互、日期和时间等内容的介绍，并结合大量的图示、举例、录像等，使读者快速掌握 PHP 语言，并为以后编程奠定坚实的基础。

第 1 章

初识 PHP

（视频讲解：19 分钟）

PHP 是一种服务器端 HTML-嵌入式脚本描述语言，其最强大和最重要的特征就是跨平台和面向对象。本章将向读者简单介绍 PHP 语言和 PHP 5 的新特性、PHP 的发展趋势以及学好 PHP 语言的方法等，其主要目的是让读者对 PHP 语言有一个整体的了解，然后再慢慢地学习具体内容，最后达到完全掌握 PHP 语言的目的。

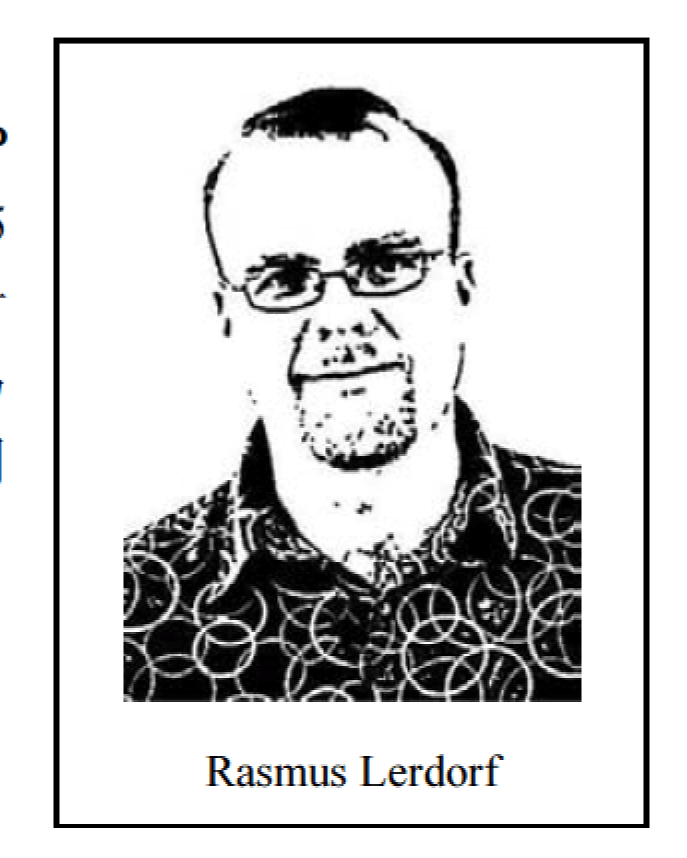
通过阅读本章，您可以：

- 了解 PHP 的发展历程及语言优势
- 了解 PHP 5 新特性
- 认识 PHP 扩展库
- 学习 PHP 网络编程的前期准备工作
- 了解 PHP 相关学习资源软件及下载网址
- 了解如何学好 PHP

1.1　PHP 概述

视频讲解：光盘\TM\lx\1\PHP 概述.exe

PHP 起源于 1995 年，由 Rasmus Lerdorf 开发，见右图。到现在，PHP 已经历了 16 年的时间洗涤，成为全球最受欢迎的脚本语言之一。由于 PHP 5 是一种面向对象的、完全跨平台的新型 Web 开发语言，所以无论从开发者角度考虑还是从经济角度考虑，都是非常实用的。PHP 语法结构简单，易于入门，很多功能只需一个函数即可实现，并且很多机构都相继推出了用于开发 PHP 的 IDE 工具、Zend 搜索引擎等新型技术。



Rasmus Lerdorf

1.1.1　什么是 PHP

PHP 是 Hypertext Preprocessor（超文本预处理器）的缩写，是一种服务器端、跨平台、HTML 嵌入式的脚本语言，其独特的语法混合了 C 语言、Java 语言和 Perl 语言的特点，是一种被广泛应用的开源式的多用途脚本语言，尤其适合 Web 开发。

PHP 是 B/S（Browser/Server 的简写，即浏览器/服务器结构）体系结构，属于三层结构。服务器启动后，用户可以不使用相应的客户端软件，只使用 IE 浏览器即可访问，既保持了图形化的用户界面，又大大减少了应用维护量。

1.1.2　PHP 语言的优势

PHP 起源于自由软件，即开放源代码软件，使用 PHP 进行 Web 应用程序的开发具有以下优势。

☑ 安全性高：PHP 是开源软件，每个人都可以看到所有 PHP 的源代码，程序代码与 Apache 编译在一起的方式也可以让它具有灵活的安全设定。PHP 具有公认的安全性能。

☑ 跨平台特性：PHP 几乎支持所有的操作系统平台（如 Win32 或 UNIX/Linux/Macintosh/FreeBSD/OS2 等），并且支持 Apache、IIS 等多种 Web 服务器，并以此广为流行。

☑ 支持广泛的数据库：可操纵多种主流与非主流的数据库，如 MySQL、Access、SQL Server、Oracle、DB2 等，其中 PHP 与 MySQL 是目前最佳的组合，它们的组合可以跨平台运行。

☑ 易学性：PHP 嵌入在 HTML 语言中，以脚本语言为主，内置丰富函数，语法简单、书写容易，方便学习掌握。

☑ 执行速度快：占用系统资源少，代码执行速度快。

☑ 免费：在流行的企业应用 LAMP 平台中，Linux、Apache、MySQL、PHP 都是免费软件，这种开源免费的框架结构可以为网站经营者节省很大一笔开支。

☑ 模板化：实现程序逻辑与用户界面分离。

☑ 支持面向对象与过程：支持面向对象和过程的两种开发风格，并可向下兼容。



☑ 内嵌 Zend 加速引擎，性能稳定快速。

1.1.3　PHP 5 的新特性

PHP 5 中的对象已经进行了较系统和全面的调整，下面着重讲述 PHP 5 中新的对象模式。

☑ 构造函数和析构函数。
☑ 对象的引用。
☑ 对象的克隆（clone）。
☑ 对象中的私有、公共及受保护模式（public/private 和 protected 关键字）。
☑ 接口（Interface）。
☑ 抽象类。
☑ __call。
☑ __set 和__get。
☑ 静态成员。

1.1.4　PHP 的发展趋势

TIOBE 世界编程语言排行榜在一定程度上体现了编程语言在当前的流行趋势。TIOBE 在编程语言排行榜 2005 年 1 月份的头条中宣布：PHP 获得 2004 年最佳编程语言称号。到 2008 年，PHP 的发展一直呈现稳步的上升趋势，从 2007 年的第 5 名上升到第 4 名，其市场占有率也在不断攀升，1～4 月份的市场占有率分别为 9.195%、9.890%、10.138%、10.328%。2007 年下半年，国外不少官方机构和著名大学都对 PHP 提供了某种程度的支持，IBM 便是其中的代表。从 2008 年 1～4 月份的排名来看，在动态语言方面，各种语言的排名基本上没有变化，PHP 目前略占优势。2008 年 4 月份 TIOBE 世界编程语言排行的相关数据说明如图 1.1 所示。

2008年4月排名	2007年4月排名	排位变化	编程语言	2008年4月流行度	自2007年4月变化	状态
1	1	=	Java	20.529%	+2.17%	A
2	2	=	C	14.684%	-0.25%	A
3	5	↑↑	(Visual) Basic	11.699%	+3.42%	A
4	4	=	PHP	10.328%	+1.69%	A
5	3	↓↓	C++	9.945%	-0.77%	A
6	6	=	Perl	5.934%	-0.10%	A
7	7	=	Python	4.534%	+0.72%	A
8	8	=	C#	3.834%	+0.28%	A
9	10	↑	Ruby	2.855%	+0.06%	A
10	11	↑	Delphi	2.665%	+0.33%	A
11	9	↓↓	JavaScript	2.434%	-0.70%	A
12	14	↑↑	D	1.169%	-0.35%	A
13	13	=	PL/SQL	0.608%	-1.28%	B
14	12	↓↓	SAS	0.572%	-1.63%	A--
15	21	↑↑↑↑↑	Pascal	0.513%	-0.06%	B
16	17	↑	Lisp/Scheme	0.476%	-0.20%	B
17	22	↑↑↑↑↑	FoxPro/xBase	0.459%	-0.09%	B
18	18	=	COBOL	0.409%	-0.24%	A--
19	16	↓↓↓	Ada	0.393%	-0.29%	B
20	31	↑↑↑↑↑↑↑↑↑↑	ColdFusion	0.384%	+0.11%	B

图 1.1　2008 年 4 月份 TIOBE 世界编程语言排行的相关数据说明

TIOBE 编程语言排行榜衡量了各种编程语言的流行程度。该排行榜每月发布一次，统计数据包括全球范围的软件工程师、培训课程以及第三方供应商，数据来自 Google、MSN 和 Yahoo!等流行搜索引擎。

说明

TIOBE 声称该排行榜并不是要评选最优秀的编程语言或是统计完成编程量最大的语言，但它可以帮助用户了解自己掌握的编程语言是否跟得上时代的发展，并且为开发新的软件系统需要对编程语言做策略性选择时提供参考。

近几年 PHP 呈现上升趋势，如图 1.2 所示，这也正说明了 PHP 语言简单、易学、面向对象和安全等特点正在被更多人所认同。相信新的 PHP 语言将会朝着更加企业化的方向迈进，并且将更适合大型系统的开发。

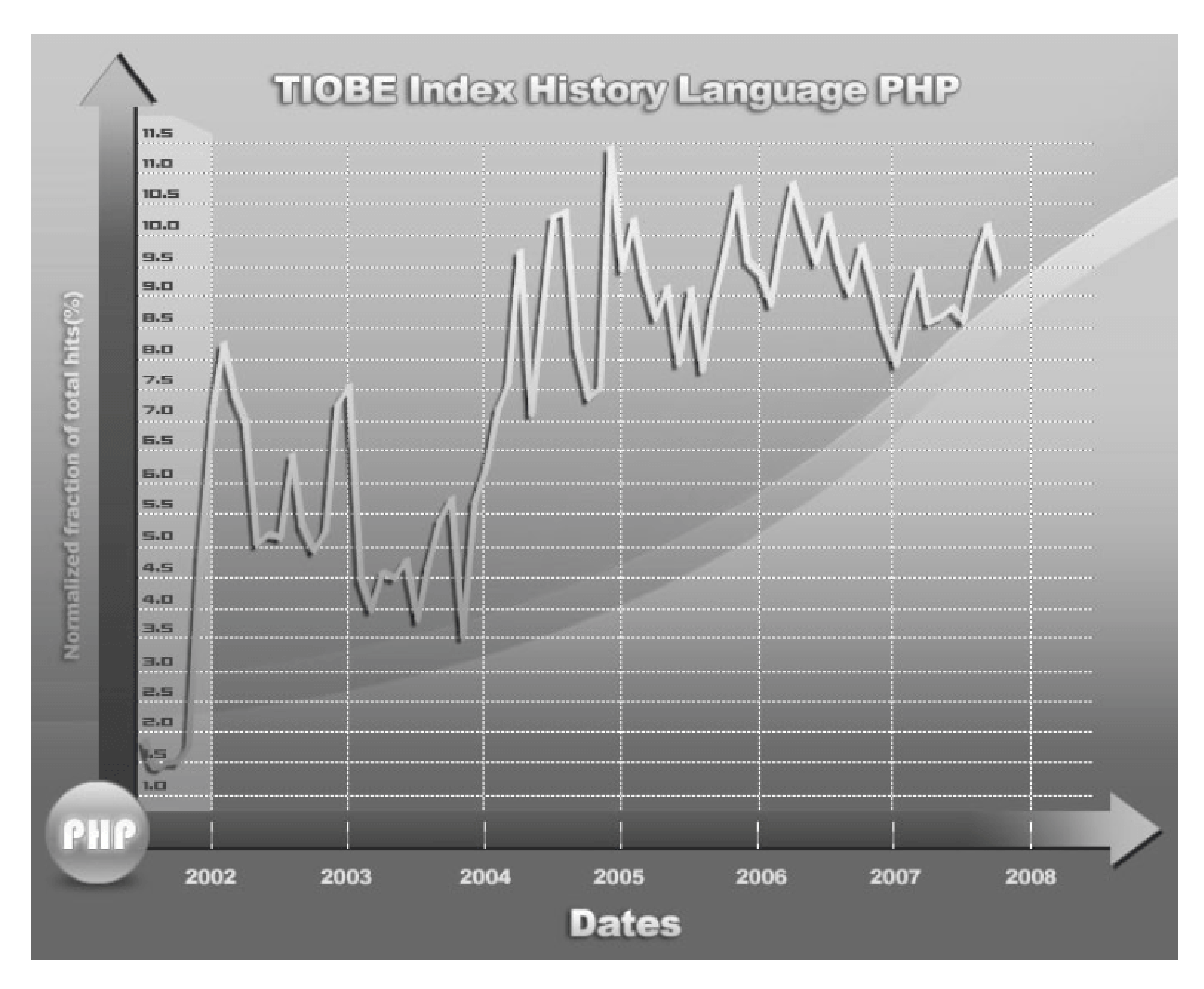


图 1.2　PHP 语言近几年的走势图

PHP 凭借其代码开源、完全免费和安全性高等特性，必将有其令人瞩目的发展前景。PHP 具有完全跨平台性，Linux+Apache+PHP+MySQL 已经成为当今建设网站的一种优良框架结构。

随着主流 PHP 5 的诞生、Zend II 引擎的采用、面向对象的支持以及模板化，PHP 编程进入了一个新时代，用户数量呈稳步上升趋势。PHP 凭借其强大的功能，在未来必将呈现出良好的发展趋势。

说明

图 1.1 和图 1.2 中的数据摘自 http://www.tiobe.com 网站。



1.1.5　PHP 的应用领域

PHP 在互联网高速发展的今天，应用范围可谓非常广泛，PHP 的应用领域主要包括：

☑ 中小型网站的开发。
☑ 大型网站的业务逻辑结果展示。
☑ Web 办公管理系统。
☑ 硬件管控软件的 GUI。
☑ 电子商务应用。
☑ Web 应用系统开发。
☑ 多媒体系统开发。
☑ 企业级应用开发。

PHP 正吸引着越来越多的 Web 开发人员。PHP 无处不在，它可应用于任何地方、任何领域，并且已拥有几百万个用户，其发展速度要快于在它之前的任何一种计算机语言。PHP 能够给企业和最终用户带来数不尽的好处。据最新数据统计，全世界有超过 2200 万的网站和 1.5 万家公司在使用 PHP 语言，包括百度、雅虎、Google、YouTube、Digg 等著名网站，也包括汉莎航空电子订票系统、德意志银行的网上银行、华尔街在线的金融信息发布系统等，甚至军队系统这类苛刻的环境也选择使用 PHP 语言。除此之外，PHP 也是企业用来构建服务导向型、创造和混合 Web 于一体新一代的综合性商业所使用的语言，成为开源商业应用发展的方向。

1.2　扩　展　库

视频讲解：光盘\TM\lx\1\扩展库.exe

PHP 5 一直在不断升级更新，总体上围绕着性能、安全与新特性，不断为开发者提供新的动力。PHP 提供了一些扩展库，这些扩展库使 PHP 如虎添翼，更加灵活方便，如网上社区、BBS 论坛等，如果没有扩展库的支持，它们都可能无法使用，因此在安装 PHP 时要根据以后的用途选择安装。

从 PHP 5 开始，PHP 即新增了内置的标准扩展库：XML 扩展库-DOM、SimpleXML、SPL、SQLite 等，而像 MySQL、MySQLi、Overload、GD2 等这些库则被放在 PECL 外部扩展库中，需要时在 php.ini 配置文件中选择加载。

在 Windows 下加载扩展库，是通过修改 php.ini 文件来完成的。用户也可以在脚本中通过使用 dl() 函数来动态加载。PHP 扩展库的 DLL 文件都具有 php_前缀。

很多扩展库都内置于 Windows 版本的 PHP 中，要加载这些扩展库不需要额外的 DLL 文件和 extension 配置指令。Windows 下的 PHP 扩展库列表列出了需要或曾经需要额外 PHP DLL 文件的扩展库。

在编辑 php.ini 文件时，应注意以下几点：

☑ 需要修改 extension_dir 设置以指向用户放置扩展库的目录或者放置 php_*.dll 文件的位置。例如：

```
extension_dir = C:\php\extensions
```



☑ 要在 php.ini 文件中启用某扩展库，需要去掉 extension=php_*.dll 前的注释符号，即将需要加载的扩展库前的分号“;”删除。例如启用 Bzip2 扩展库，需要将下面这行代码：

```
;extension=php_bz2.dll
```

改成：

```
extension=php_bz2.dll
```

☑ 有些扩展库需要额外的 DLL 才能工作，其中一部分 DLL 文件包括在发行包中（PHP 5 中在主目录下），但还有一些，如 Oracle（php_oci8.dll）所需要的 DLL 没有绑定在发行包中。如果安装 PHP 5，需将绑定的 DLL 从 C:\php 5\dlls 复制到主目录 C:\php 中。值得注意的是，必须将 C:\php 5 放到系统路径 PATH 中。

☑ 某些 DLL 没有绑定在 PHP 发行包中。PECL 中有日益增加、数目巨大的 PHP 扩展库，这些扩展库需要单独下载。

注意

如果运行服务器模块版本的 PHP，在修改了 php.ini 之后应注意重新启动 Web 服务器，使改动生效。

PHP 内置扩展库列表，如表 1.1 所示。

表 1.1 PHP 内置扩展库列表

扩 展 库	说 明	注 解
php_bz2.dll	Bzip2 压缩函数库	无
php_calendar.dll	历法转换函数库	自 PHP 4.0.3 起内置
php_cpdf.dll	ClibPDF 函数库	无
php_crack.dll	密码破解函数库	无
php_ctype.dll	ctype 家族函数库	自 PHP 4.3.0 起内置
php_curl.dll	CURL，客户端 URL 函数库	需要 libeay32.dll，ssleay32.dll（已附带）
php_cybercash.dll	网络现金支付函数库	PHP<=4.2.0
php_dba.dll	DBA，数据库（dbm 风格）抽象层函数库	无
php_dbase.dll	dBase 函数库	无
php_dbx.dll	dbx 函数库	
php_domxml.dll	DOM XML 函数库	PHP<=4.2.0 需要 libxml2.dll（已附带），PHP>=4.3.0 需要 iconv.dll（已附带）
php_dotnet.dll	.NET 函数库	PHP<=4.1.1
php_exif.dll	EXIF 函数库	需要 php_mbstring.dll，并且在 php.ini 中，php_exif.dll 必须在 php_mbstring.dll 之后加载
php_fbsql.dll	FrontBase 函数库	PHP<=4.2.0
php_fdf.dll	FDF：表单数据格式化函数库	需要 fdftk.dll（已附带）
php_filepro.dll	filePro 函数库	只读访问
php_ftp.dll	FTP 函数库	自 PHP 4.0.3 起内置

续表

扩　展　库	说　　明	注　　解
php_gd.dll	GD 库图像函数库	在 PHP 4.3.2 中删除。此外注意在 GD1 中不能用真彩色函数，应用 php_gd2.dll 替代
php_gd2.dll	GD2 库图像函数库	GD2
php_gettext.dll	Gettext 函数库	PHP<=4.2.0 需要 gnu_gettext.dll（已附带），PHP>=4.2.3 需要 libintl-1.dll，iconv.dll（已附带）
php_hyperwave.dll	HyperWave 函数库	无
php_iconv.dll	ICONV 字符集转换	需要 iconv-1.3.dll（已附带），PHP>=4.2.1 需要 iconv.dll
php_ifx.dll	Informix 函数库	需要 Informix 库
php_iisfunc.dll	IIS 管理函数库	无
php_imap.dll	IMAP、POP3 和 NNTP 函数库	无
php_ingres.dll	Ingres II 函数库	需要 Ingres II 库
php_interbase.dll	InterBase functions	需要 gds32.dll（已附带）
php_java.dll	Java 函数库	PHP<=4.0.6 需要 jvm.dll（已附带）
php_ldap.dll	LDAP 函数库	PHP<=4.2.0 需要 libsasl.dll（已附带），PHP>=4.3.0 需要 libeay32.dll，ssleay32.dll（已附带）
php_mbstring.dll	多字节字符串函数库	无
php_mcrypt.dll	Mcrypt 加密函数库	需要 libmcrypt.dll
php_mhash.dll	Mhash 函数库	PHP>=4.3.0 需要 libmhash.dll（已附带）
php_mime_magic.dll	Mimetype 函数库	需要 magic.mime（已附带）
php_ming.dll	Ming 函数库（Flash）	无
php_msql.dll	mSQL 函数库	需要 msql.dll（已附带）
php_mssql.dll	MSSQL 函数库	需要 ntwdblib.dll（已附带）
php_mysql.dll	MySQL 函数库	PHP>=5.0.0 需要 libmysql.dll（已附带）
php_mysqli.dll	MySQLi 函数库	PHP>=5.0.0 需要 libmysql.dll(PHP<=5.0.2 中是 libmysqli.dll）（已附带）
php_oci8.dll	Oracle 8 函数库	需要 Oracle 8.1+客户端库
php_openssl.dll	OpenSSL 函数库	需要 libeay32.dll（已附带）
php_oracle.dll	Oracle 函数库	需要 Oracle 7 客户端库
php_overload.dll	对象重载函数库	自 PHP 4.3.0 起内置
php_pdf.dll	PDF 函数库	无
php_pgsql.dll	PostgreSQL 函数库	无
php_printer.dll	打印机函数库	无
php_shmop.dll	共享内存函数库	无
php_snmp.dll	SNMP 函数库	仅用于 Windows NT
php_soap.dll	SOAP 函数库	PHP>=5.0.0
php_sockets.dll	Socket 函数库	无
php_sybase_ct.dll	Sybase 函数库	需要 Sybase 客户端库
php_tidy.dll	Tidy 函数库	PHP>=5.0.0
php_tokenizer.dll	Tokenizer 函数库	自 PHP 4.3.0 起内置
php_w32api.dll	W32api 函数库	无

续表

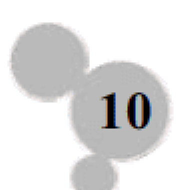
扩　展　库	说　　明	注　　解
php_xmlrpc.dll	XML-RPC 函数库	PHP>=4.2.1 需要 iconv.dll（已附带）
php_xslt.dll	XSLT 函数库	PHP<=4.2.0 需要 sablot.dll，expat.dll（已附带）PHP>=4.2.1 需要 sablot.dll，expat.dll，iconv.dll（已附带）
php_yaz.dll	YAZ 函数库	需要 yaz.dll（已附带）
php_zip.dll	Zip 文件函数库	只读访问
php_zlib.dll	ZLib 压缩函数库	自 PHP 4.3.0 起内置

1.3　如何学好 PHP

视频讲解：光盘\TM\lx\1\如何学好 PHP.exe

怎样学好 PHP 语言，这是所有初学者共同面临的问题，其实，每种语言的学习方法都大同小异，需要注意的有以下几点：

- ☑ 明确自己的学习目标和大的方向，选择并锁定一门语言，按照自己的学习方向努力学习、认真研究。
- ☑ 学会配置 PHP 的开发环境，选择一种适合自己的开发工具。
- ☑ 扎实的基础对于一个程序员来说尤为重要，因此建议读者多阅读一些基础教材，了解基本的编程知识，掌握常用的函数。
- ☑ 了解设计模式。开发程序必须编写程序代码，这些代码必须具有高度的可读性，这样才能使编写的程序具有调试、维护和升级的价值，学习一些设计模式，就能更好地把握项目的整体结构。
- ☑ 多实践，多思考，多请教。不要死记语法，在刚接触一门语言，特别是学习 PHP 语言时，掌握好基本语法，反复实践。仅读懂书本中的内容和技术是不行的，必须动手编写程序代码，并运行程序、分析运行结构，让大脑对学习内容有个整体的认识和肯定。用自己的方式去思考问题、编写代码来提高编程思想。平时可以多借鉴网上一些好的功能模块，培养自己的编程思想。多向他人请教，学习他人的编程思想。多与他人沟通技术问题，提高自己的技术和见识。这样才可以快速地进入学习状态。
- ☑ 学技术最忌急躁，遇到技术问题，必须冷静对待，不要让自己的大脑思绪紊乱，保持清醒的头脑才能分析和解决各种问题。可以尝试听歌、散步、玩游戏等活动放松自己。遇到问题，还要尝试自己解决，这样可以提高自己的程序调试能力，并对常见问题有一定的了解，明白出错的原因，进而举一反三，解决其他关联的错误问题。
- ☑ PHP 函数有几千种，需要下载一个 PHP 中文手册和 MySQL 手册，或者查看 PHP 函数类的相关书籍，以便解决程序中出现的问题。
- ☑ 现在很多 PHP 案例书籍都配有视频录像，可以看一些视频录像领悟他人的编程思想。只有掌握了整体的开发思路之后，才能够系统地学习编程。
- ☑ 养成良好的编程习惯。



☑ 遇到问题不要放弃，要有坚持不懈、持之以恒的精神。

1.4 学 习 资 源

视频讲解：光盘\TM\lx\1\PHP 的学习资源.exe

下面为读者推荐一些学习 PHP 的相关资源。使用这些资源，可以帮助读者找到精通 PHP 的捷径。

1.4.1 常用软件资源

1. PHP 开发工具

PHP 的开发工具很多，常用的开发工具有 Dreamweaver、ZendStudio 和 EditPlus2，还有最新的 Delphi for PHP。每个开发工具各有优势，一个好的开发工具往往会达到事半功倍的效果，读者可根据自己的需求选择使用。

开发工具下载网站为 http://www.newhua.com/或 http://www.skycn.com/。

2. 下载 PHP 用户手册

学习 PHP 语言，配备一个 PHP 参考手册是必要的，就像我们在学习汉字时手中必须具备一本新华字典一样。PHP 参考手册对 PHP 的函数进行了详细的讲解和说明，并且还给出了一些简单的示例，同时还对 PHP 的安装与配置、语言参考、安全和特点等内容进行了介绍。

在 http://www.php.net/docs.php 网站上，提供有 PHP 的各种语言、格式和版本的 PHP 参考手册，读者可以进行在线阅读，也可以下载。

PHP 参考手册不但对 PHP 的函数进行了解释和说明，而且还提供了快速查找的方法，让用户可以更加方便地查找到指定的函数。PHP 参考手册下载版如图 1.3 所示。

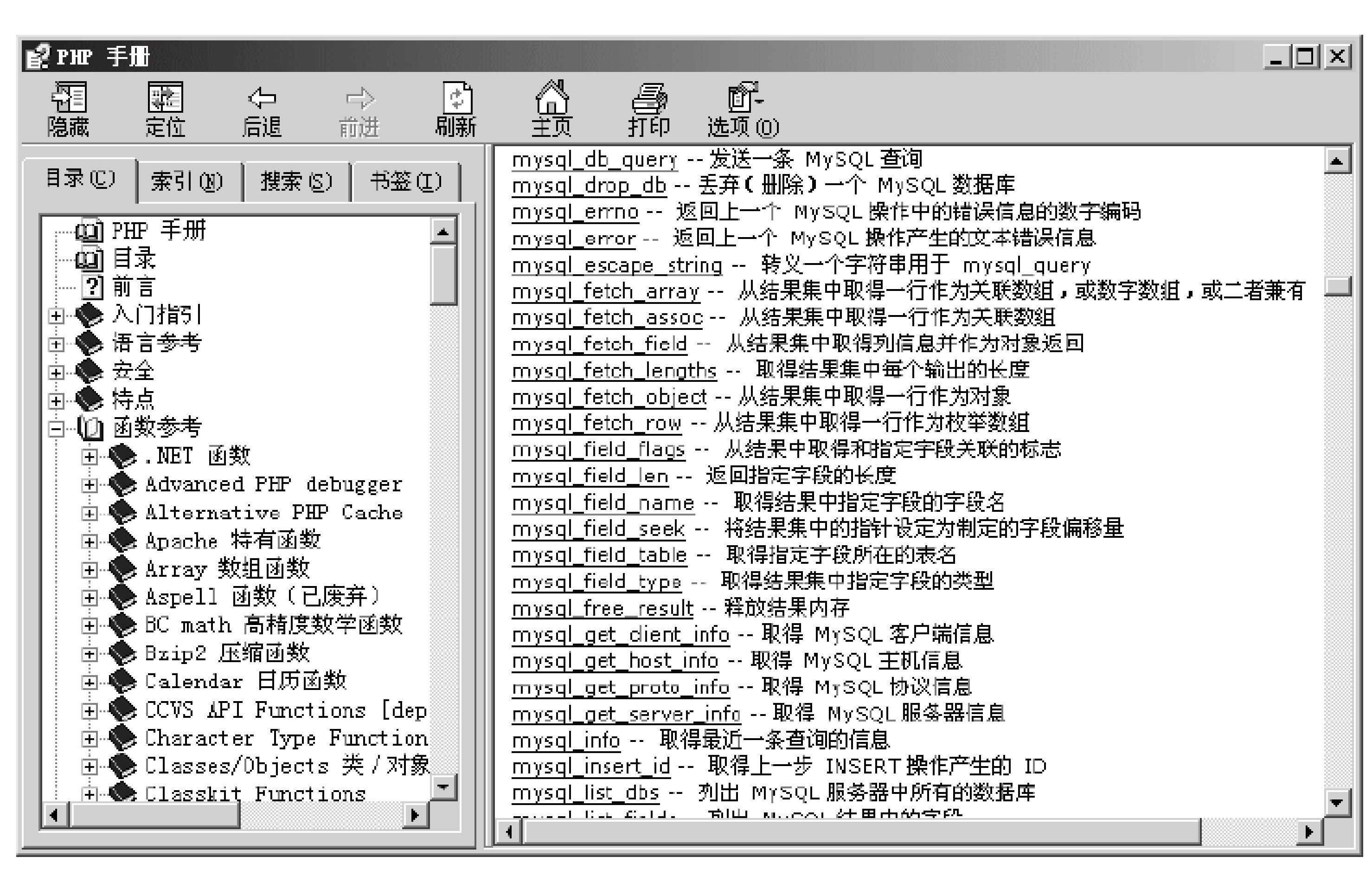


图 1.3　PHP 参考手册

1.4.2 常用网上资源

下面提供一些大型的 PHP 技术论坛和社区，这些资源不但可以提高 PHP 编程者的技术水平，也是程序员学习和工作的好帮手。

1. PHP 技术论坛

☑ PHP100

http://www.php100.com

☑ PHP 中国

http://www.phpchina.com

2. PHP 论坛

http://www.php.cn

1.4.3 主要图书网站

下面提供一些国内比较大的 PHP 图书网站，内容丰富、信息全面、查阅方便，是读者了解 PHP 图书信息的窗口。

☑ 当当网

http://www.dangdang.com.cn

☑ 卓越网

http://www.amazon.com.cn

☑ 互动出版网

www.china-pub.com

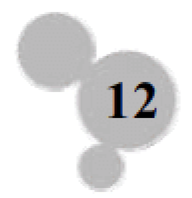
☑ 明日图书网

http://www.mingribook.com

1.5 网站建设的基本流程

视频讲解：光盘\TM\lx\1\网站建设的基本步骤.exe

建立一个网站是需要特定工作流程的。本节将介绍网站建设的基本流程，从而使读者在明确开发流程的基础上，能够更顺利地进行网站开发工作。网站建设的基本流程如图 1.4 所示。



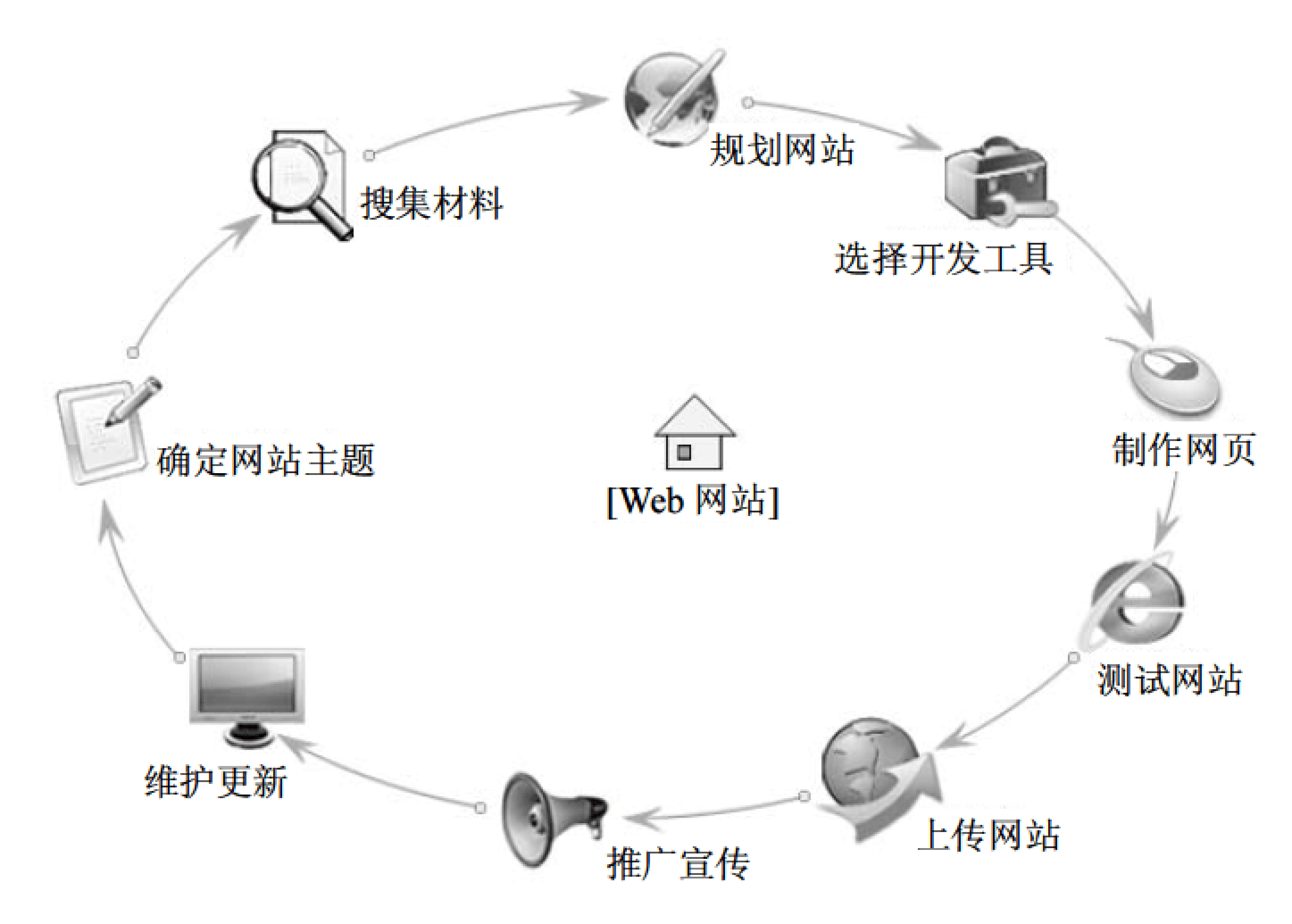


图 1.4　网站建设的基本流程

1.6 小　　结

本章重点讲述了 PHP 的发展历程及语言优势，详细介绍了 PHP 5 的新增功能，随后还介绍了 PHP 5 的扩展库。在学习 PHP 之前，先学习了一些术语与专有名词。最后学习了构建网站的基本流程与 PHP 相关资源的获取路径。

第 2 章

PHP 环境搭建和开发工具

（视频讲解：52 分钟）

要使用 PHP，首先要建立 PHP 开发环境。本章将介绍两种操作系统（Windows 和 Linux）下的 PHP 环境搭建及流行的开发工具。此外还为初学者介绍了几种 PHP 组合包来简化安装过程。最后，使用 Dreamweaver 开发 PHP 第一个实例。

通过阅读本章，您可以：

- 掌握在 Windows 下使用 AppServ 组合包配置 PHP 开发环境
- 掌握配置 IIS+PHP+MySQL 搭建 PHP 开发环境
- 了解搭建 Linux 下的 PHP 环境
- 了解 PHP 常用开发工具
- 了解第一个 PHP 实例

2.1 在 Windows 下使用 AppServ 组合包

视频讲解：光盘\TM\lx\2\Windows 下 AppServ 组合包的安装.exe

组合包，就是将 Apache、PHP、MySQL 等服务器软件和工具安装配置完成后打包处理。开发人员只要将已配置的套件解压到本地硬盘中即可使用，无须再另行配置。组合包实现了 PHP 开发环境的快速搭建。对于刚开始学习 PHP 的程序员，建议采用此方法搭建 PHP 的运行环境。虽然在灵活性上要差很多，但组合包安装简单、速度较快、运行稳定。

目前网上流行的组合包有十几种，安装基本都是大同小异。这里推荐 3 种组合包：EasyPHP、AppServ 和 XAMPP。

建议新手使用 EasyPHP 或 AppServ，它们都是 Apache+MySQL+PHP 开发环境的。而 XAMPP 则相对要复杂一些，不仅可以切换 PHP 4 和 PHP 5，还集成 Perl 开发环境、第三方扩展库等，并且 XAMPP 对开发平台进行了优化和整理。如果对 PHP 的开发环境已经有了一定的了解，则推荐读者使用 XAMPP。

- ☑ EasyPHP 下载地址：http://www.easyphp.org/。
- ☑ AppServ 下载地址：http://www.AppServnetwork.com/。
- ☑ XAMPP 下载地址：http://www.Apachefriends.org/。

注意

想要安装这些组合包，必须保证系统中没有安装 Apache、PHP 和 MySQL。否则，需要先将这些软件卸载后，再开始安装组合包。组合包的安装很简单，只要将程序解压或安装到指定目录后即可直接使用。

应用 AppServ 集成化安装包搭建 PHP 开发环境的操作步骤如下：

（1）双击 AppServ-win32-2.5.7.exe 文件，打开如图 2.1 所示的 AppServ 启动页面。

（2）单击 Next 按钮，打开如图 2.2 所示的 AppServ 安装协议页面。

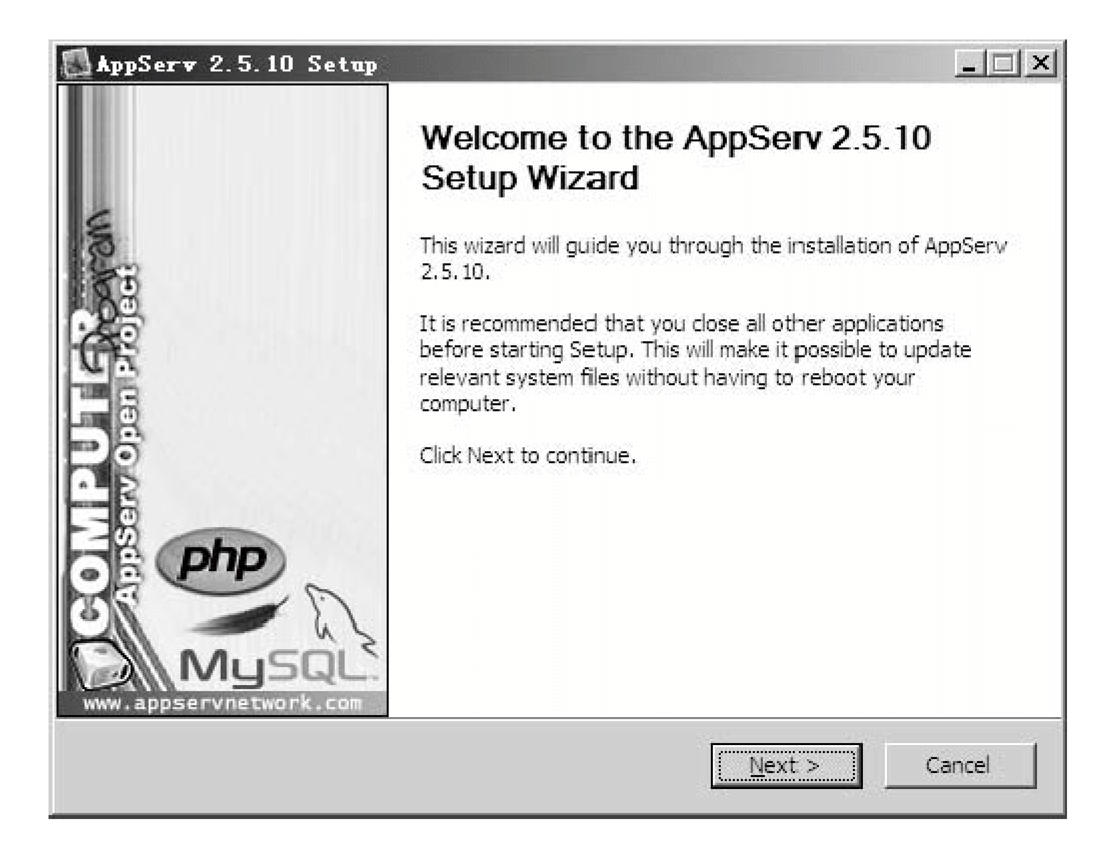


图 2.1 AppServ 启动页面

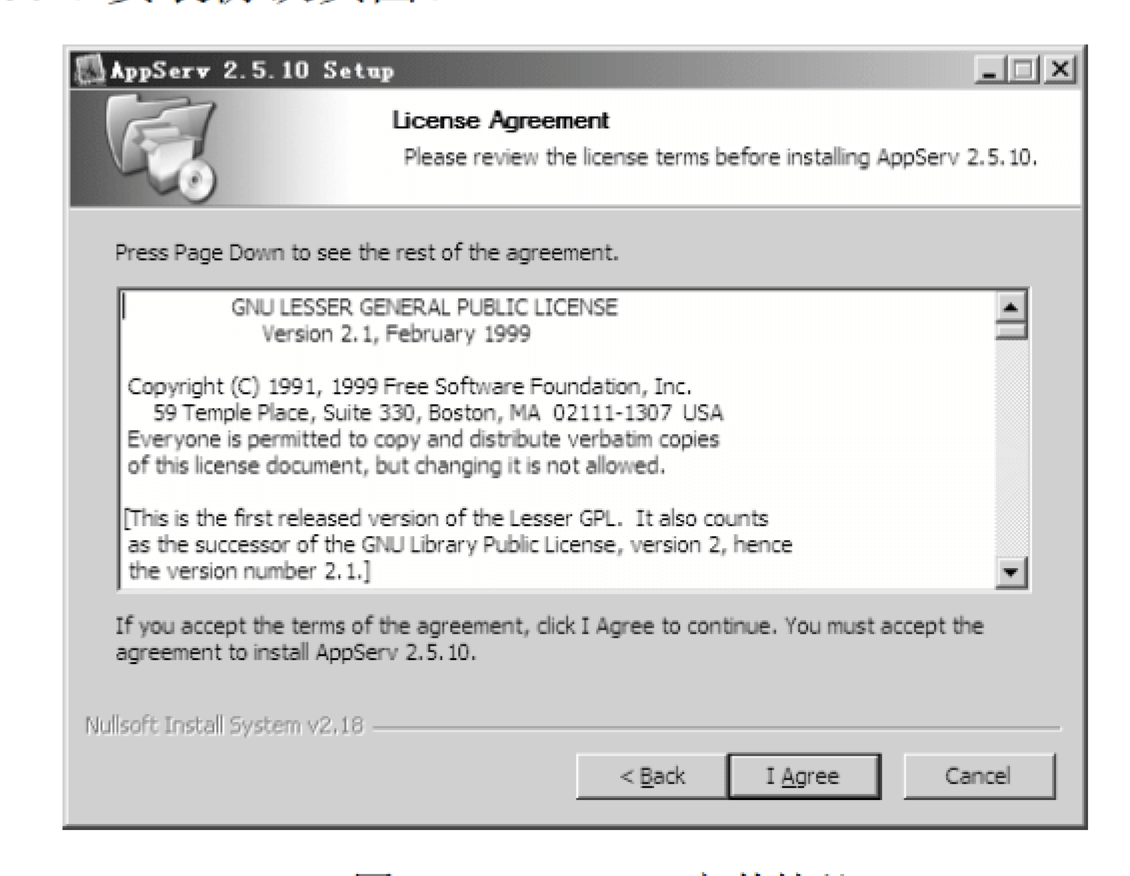
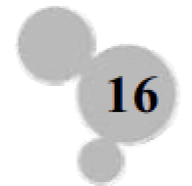


图 2.2 AppServ 安装协议



（3）单击 I Agree 按钮，打开如图 2.3 所示的页面。在该页面中可以设置 AppServ 的安装路径（默认安装路径一般为 C:\AppServ），AppServ 安装完成后，Apache、MySQL、PHP 都将以子目录的形式存储到该目录下。

（4）单击 Next 按钮，打开如图 2.4 所示的页面，在该页面中可以选择要安装的程序和组件（默认为全选状态）。

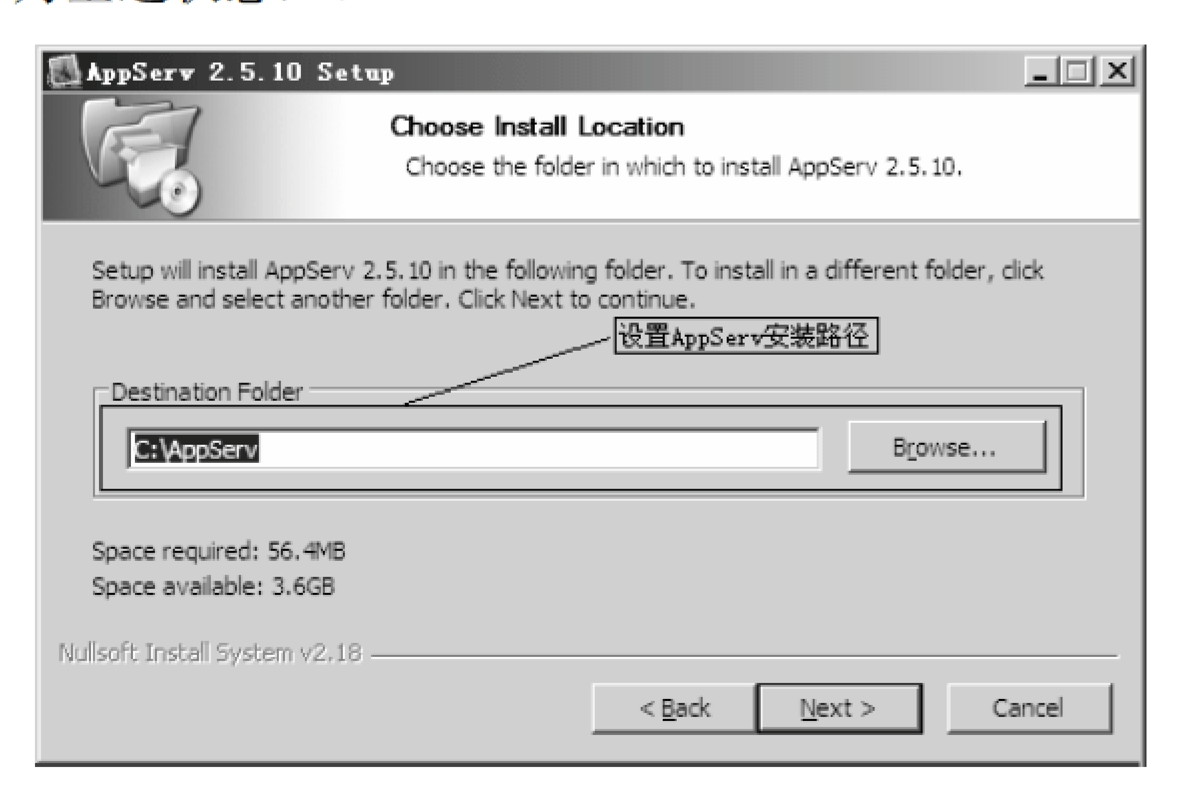


图 2.3　AppServ 安装路径选择

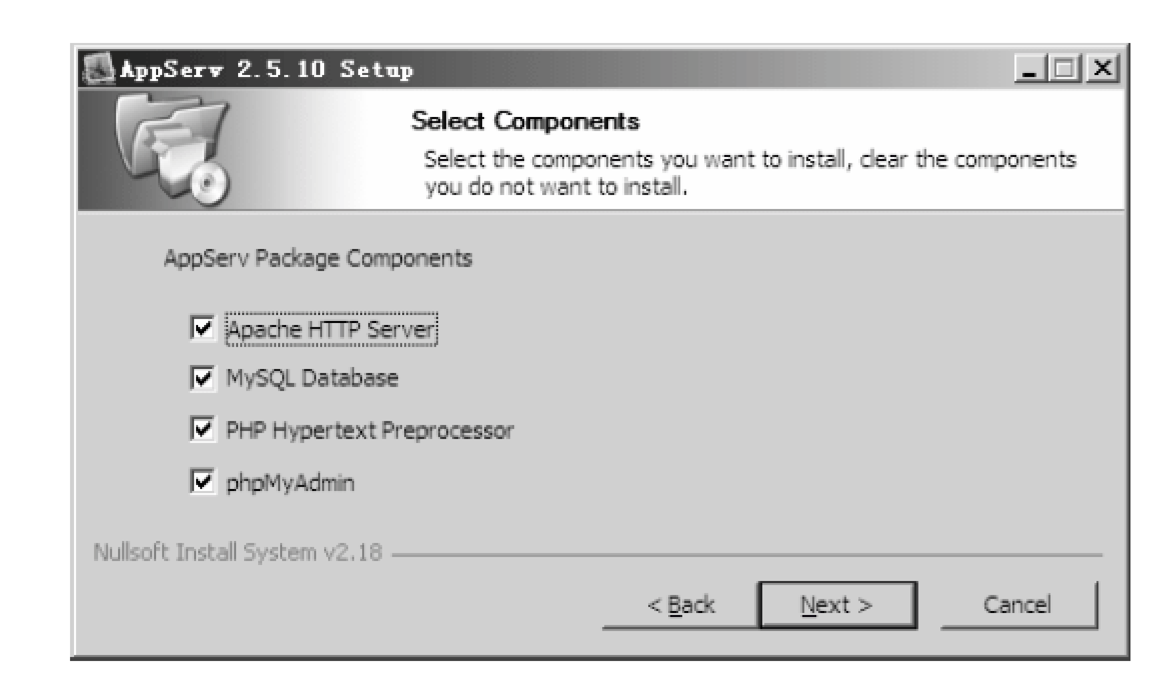


图 2.4　AppServ 安装选项

（5）单击 Next 按钮，打开如图 2.5 所示的页面，该页面主要设置 Apache 的端口号。

注意

服务器端口号的设置至关重要，它直接关系到 Apache 服务器是否能够启动成功。如果本机中的 80 端口被 IIS 或者迅雷占用，那么这里仍然使用 80 端口就不能完成服务器的配置。可通过修改这里的端口（例如，改为 82），或者将 IIS 或者迅雷的端口进行修改，即可解决该问题。

（6）单击 Next 按钮，打开如图 2.6 所示的页面。该页面主要对 MySQL 数据库的 root 用户的登录密码及字符集进行设置，这里将字符集设置为“GB2312 Simplified Chinese”，表示 MySQL 数据库的字符集将采用简体中文形式。

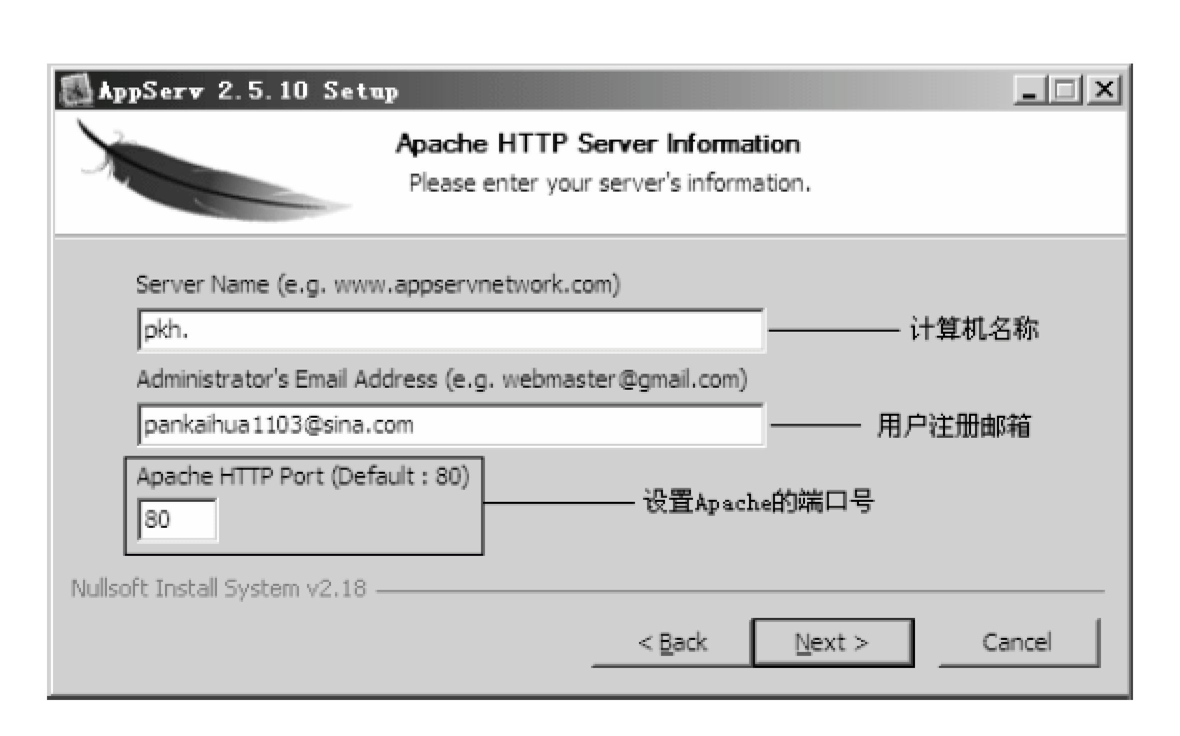


图 2.5　Apache 端口号设置

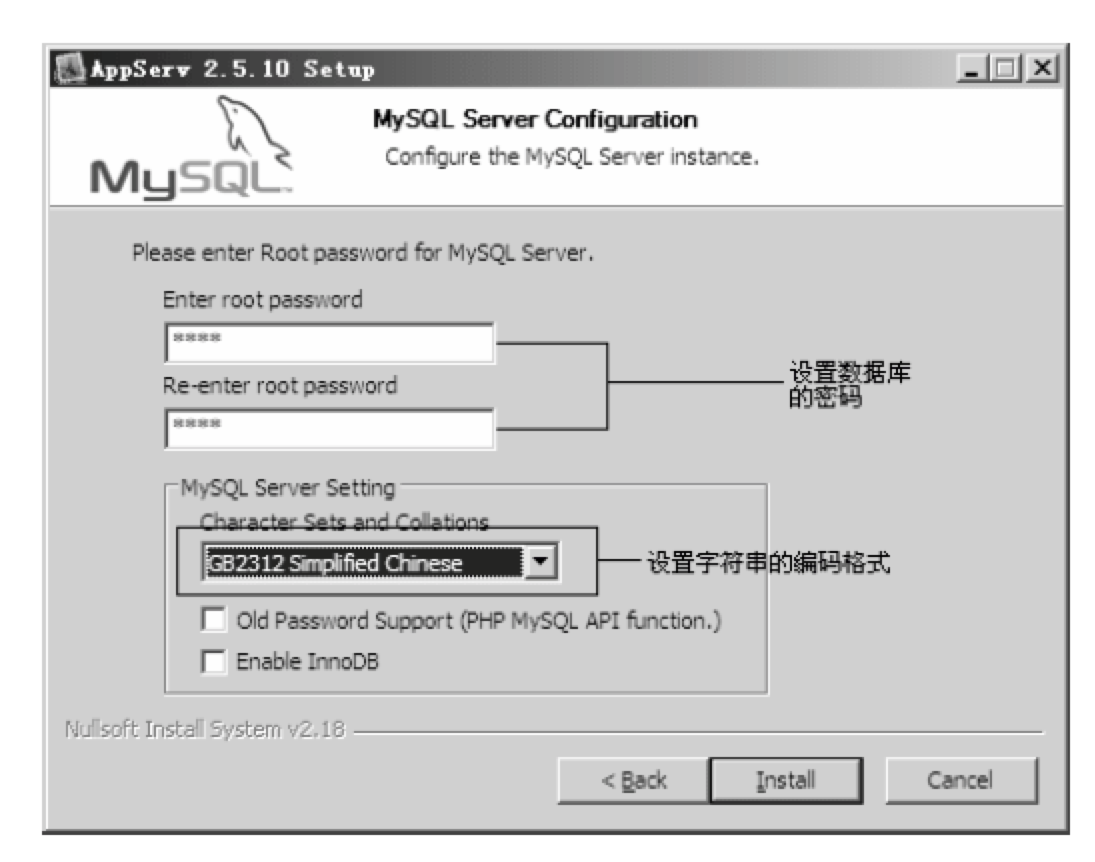


图 2.6　MySQL 设置



技巧

这里设置组合包中安装的 MySQL 数据库的密码。在设置时一定要牢记设置的密码，因为以后在程序中连接数据库时要使用到。建议读者将密码设置为 root，因为这是开发本书中的程序时所使用的数据库密码。

（7）单击 Install 按钮后开始安装，如图 2.7 所示。

（8）至此，AppServ 安装成功，如图 2.8 所示。

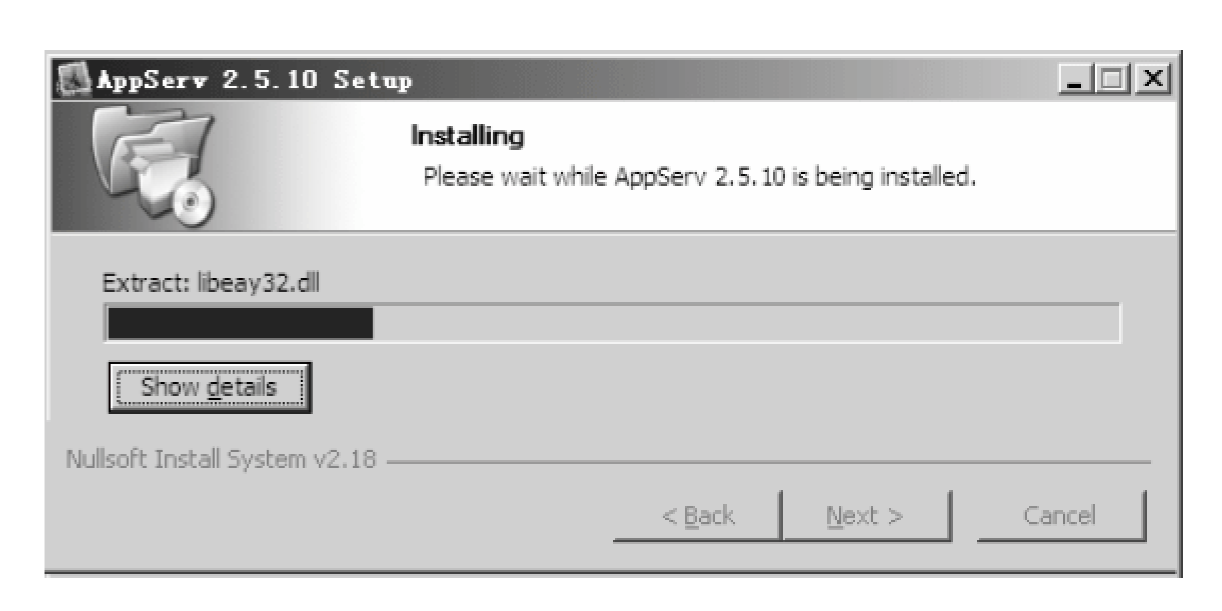


图 2.7　AppServ 安装页面

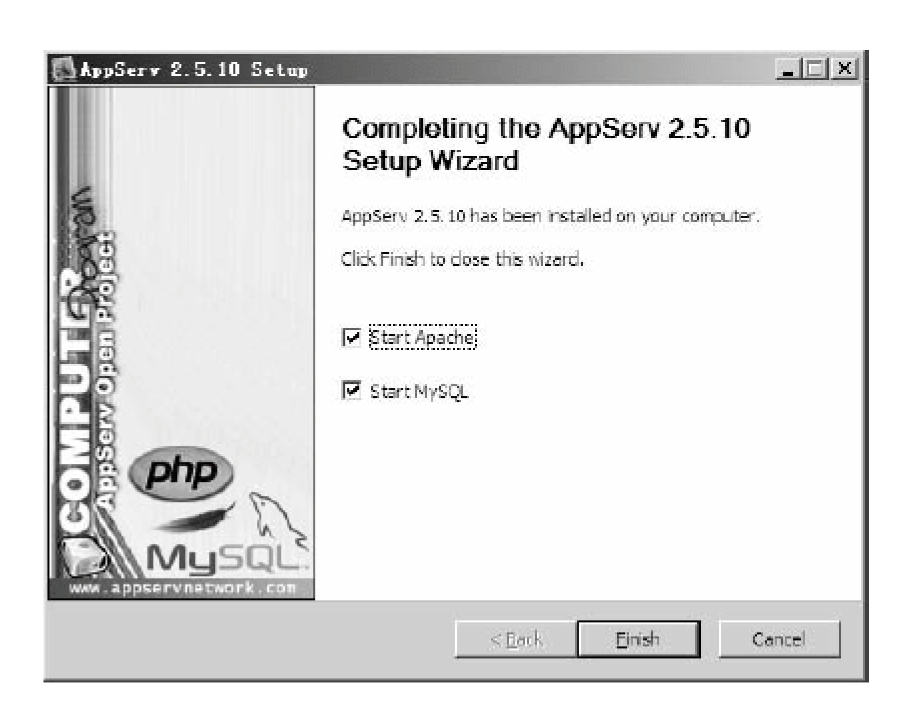


图 2.8　AppServ 安装完成页面

（9）安装好 AppServ 之后，整个目录默认安装在“C:\AppServ”路径下，此目录下包含 4 个子目录，如图 2.9 所示，用户可以将所有网页文件存放到“www”目录下。

（10）打开浏览器，在地址栏中输入“http://localhost/”或者“http://127.0.0.1/”，如果打开如图 2.10 所示的页面，则说明 AppServ 安装成功。

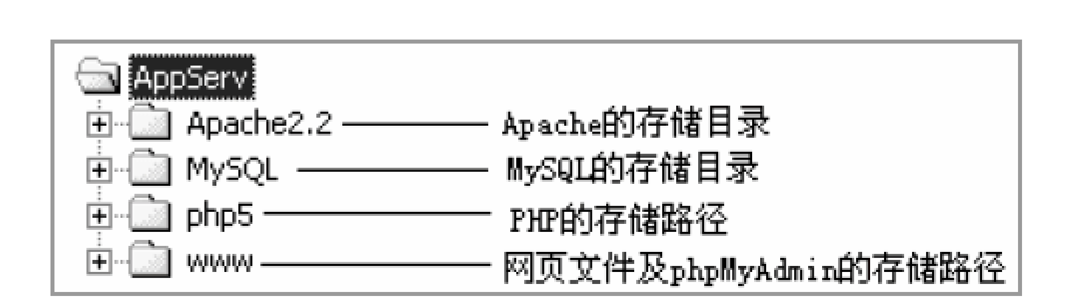


图 2.9　AppServ 目录结构

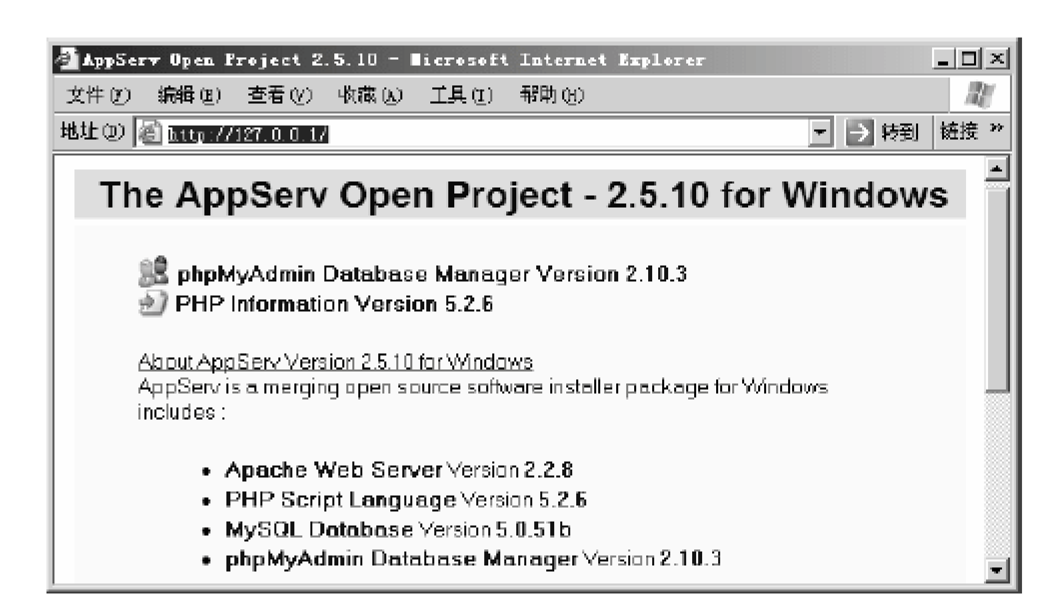
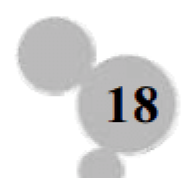


图 2.10　AppServ 测试页

2.2　在 Windows 下使用 IIS+PHP+MySQL 搭建 PHP 环境

视频讲解：光盘\TM\lx\2\Windows 下使用 IIS+PHP+MySQL 搭建 PHP 环境.exe

Internet 信息服务器即 Internet Information Server，缩写为 IIS，是 Microsoft 的 Web 服务器。它集成于 Windows NT Server 之中，方便易用，为 Web 应用程序提供了功能强大的运行平台。



2.2.1　安装 PHP 5

安装 PHP 5 的操作步骤如下：

（1）将 PHP 5 的安装文件 php-5.1.1-Win32.zip 解压到相应目录。如 C:\php、F:\php 等。这里将 PHP 放置在 F:\php 目录下。

（2）将 F:\php\目录下的 libMySQL.dll 文件复制到 C:\windows\system32\（如果是 Windows 2000，则为 C:\WINNT\system32\）目录下。

（3）将 F:\php\目录下的 php.ini-dist 文件复制到 C:\windows\（如果是 Windows 2000，则为 C:\WINNT\）目录下。把 php.ini-dist 重命名为 php.ini。

（4）打开 php.ini 文件并定位到“extension_dir = "./"”这一行，修改为 extension_dir=" F:\php\ext "。

（5）定位到“;extension=php_MySQL.dll”这一行，将前面的分号“;”去掉。这样，PHP 即可支持 MySQL 数据库。如果想要加载其他动态库，其方法是相同的。

（6）保存配置。

2.2.2　安装配置 IIS 服务器

这里以 Windows 2003 中使用的 IIS 6.0 为例来讲解 IIS 的安装。至于在其他操作系统中其安装步骤是相同的，只是使用的版本有所不同。

（1）选择“开始”/“设置”/“控制面板”命令，在弹出的界面中单击“添加/删除程序”按钮。单击左侧的“添加/删除 Windows 组件”按钮，将弹出“Windows 组件向导”窗口，选中“应用程序服务器”复选框，单击“详细信息”按钮，将弹出“应用程序服务器”窗口，选中“Internet 信息服务（IIS）”复选框，单击“详细信息”按钮，将弹出“Internet 信息服务（IIS）”窗口，选中“Internet 信息服务管理器”复选框，单击“确定”按钮，开始 IIS 信息服务管理器的安装。其操作过程如图 2.11 所示。



图 2.11　Internet 信息服务（IIS）

（2）开始安装组件，应注意安装过程中需要将系统盘放到光驱中。

（3）安装完毕后，即可使用 IIS。



技巧

如果之前没有安装 Apache 服务器，或者 Apache 服务器的端口已经修改，那么只需打开 IE 浏览器，在地址栏中输入“http://localhost”，即可看到 IIS 运行界面。如果 Apache 的端口号为 80，那么就需要修改 IIS 中的端口。具体操作可参考步骤（6）。

（4）配置 IIS。选择“开始”/“设置”/“控制面板”命令，在弹出的窗口中双击“管理工具”图标，在弹出的窗口中选择刚添加的“Internet 信息服务（IIS）管理器”，单击该图标，将弹出 IIS 管理器界面，如图 2.12 所示。

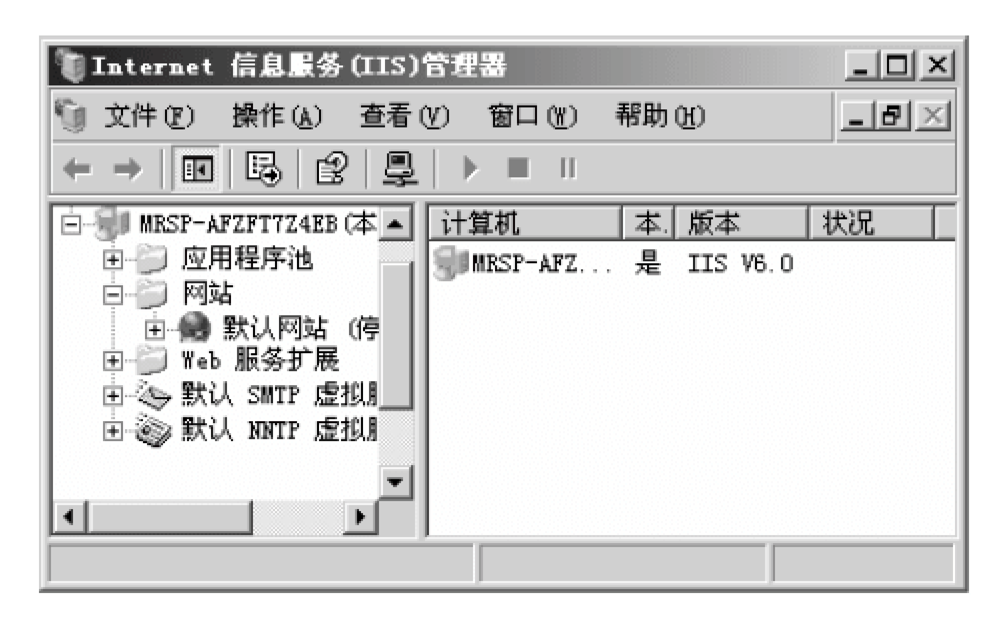


图 2.12　IIS 管理器界面

（5）图中的默认网站即为 IIS 服务器的文件目录。右击“默认网站”选项，在弹出的快捷菜单中选择“属性”命令，然后对网站的属性进行设置，如图 2.13 所示。

（6）默认选中“网站”选项卡，这里主要设置 TCP 端口号。将“TCP 端口”设置为 8000。设置完成后选择“主目录”选项卡。

（7）“主目录”选项卡中需要设置的位置主要有两处：第一处是本地路径，用户可以自行定义，如 D:\htdocs\；第二处是设置执行权限，设置为纯脚本。设置后如图 2.14 所示。

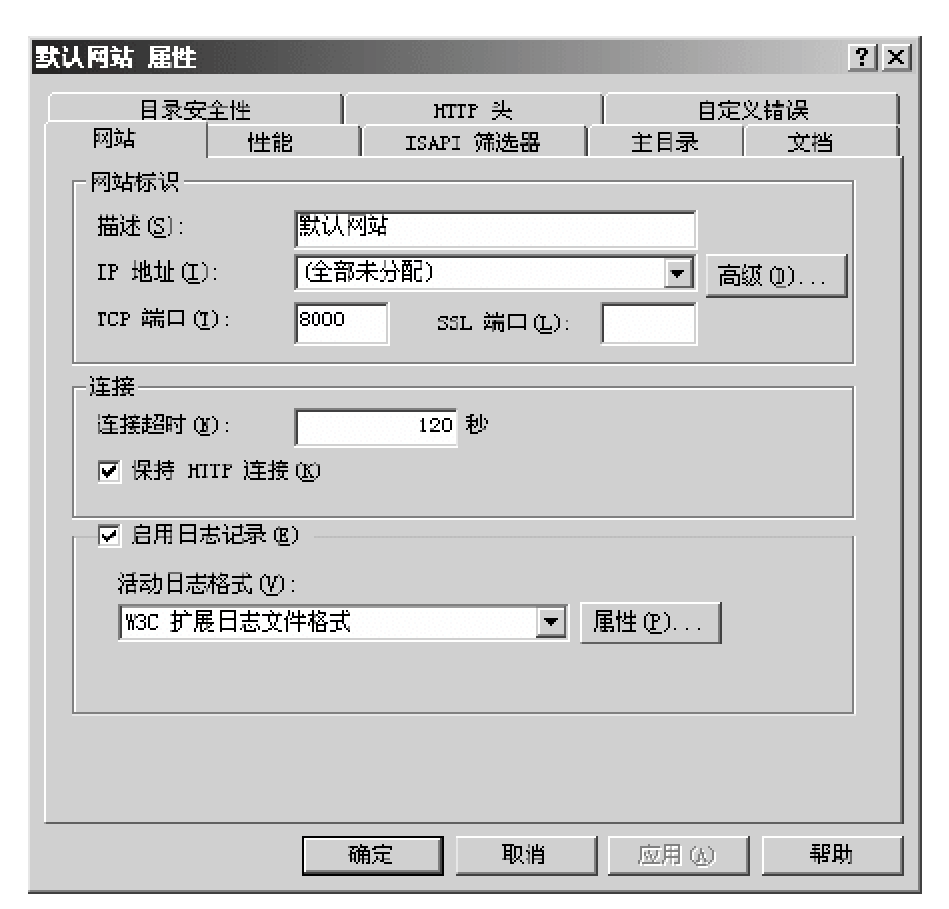


图 2.13　“网站”选项卡

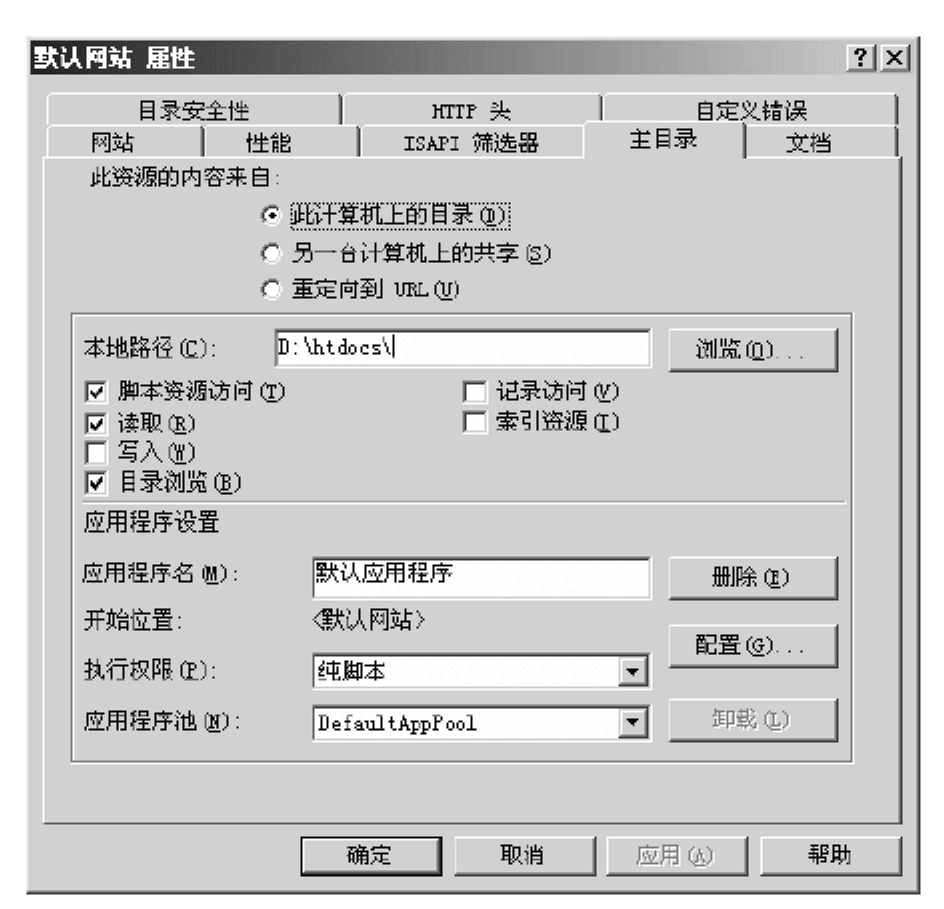


图 2.14　“主目录”选项卡



（8）单击该选项卡中的“配置”按钮，将弹出“应用程序配置”对话框，如图 2.15 所示。

（9）在“映射”选项卡中可以添加、编辑应用程序扩展。单击“添加”按钮添加如图 2.16 所示的信息，设置完毕后单击“确定”按钮返回。



图 2.15　应用程序配置



图 2.16　添加 PHP 扩展

（10）选择“文档”选项卡，添加一个.php 的默认页。添加后的页面如图 2.17 所示。

（11）关闭“默认网站 属性”对话框，在 IIS 管理器中选择“Web 服务扩展”。在右侧窗口中选择“所有未知 ISAPI 扩展”服务，单击左侧的“允许”按钮启动该服务，如图 2.18 所示。

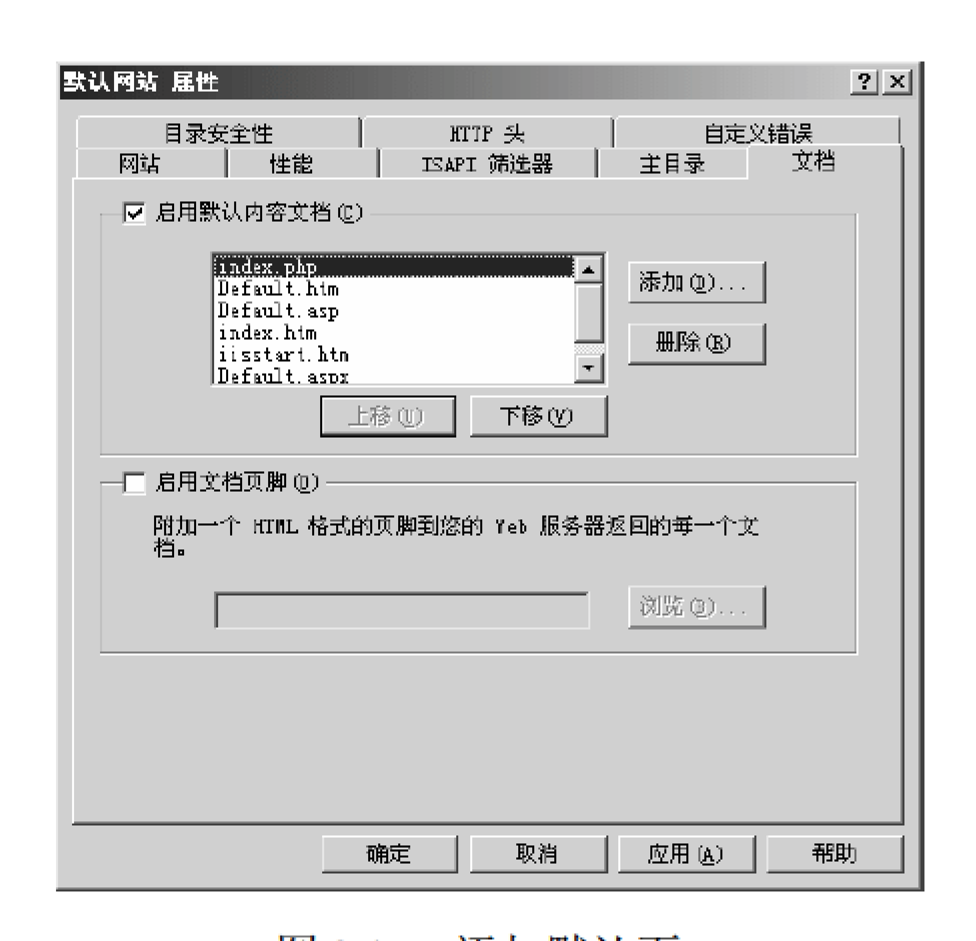




图 2.17　添加默认页

图 2.18　开启 ISAPI 扩展

（12）关闭所有界面，重启 IIS 服务器，至此设置完成。

注意

如果使用 IIS 服务器配置 PHP 的开发环境，那么就不要再使用集成化的安装包来配置。因为它们会发生冲突导致配置不能生效。

2.2.3 安装 MySQL

MySQL 是一款广受欢迎的数据库，由于开源所以市场占有率高，备受 PHP 开发者的青睐，一直被认为是 PHP 的最佳搭档。本节就来介绍 MySQL 数据库的安装过程。

说明

在学习 MySQL 之前几乎不会接触到数据库知识，也使用不到 MySQL 数据库，因此读者可以暂时不用安装 MySQL 而专心地学习 PHP 知识。

MySQL 数据库的安装步骤如下：

（1）双击 MySQL 安装文件 mysql-essential-5.0.51-win32.msi，进入欢迎界面。单击 Next 按钮，进入 Setup Type 界面。

（2）该界面中的第一项是典型安装，第二项是全部安装。这两种安装方式的路径不能改变，默认是 E:\Program Files\MySQL\MySQL Server5.0\（E 盘为系统盘）。第三项 Custom 则可以让用户自定义选择安装组件和安装路径。这里选择第三项。Setup Type 界面如图 2.19 所示。

（3）单击 Next 按钮进入到 Custom Setup 界面。选择需要安装的组件，并单击 Change 按钮来选择要安装的目录。选择完毕后单击 Next 按钮继续。Custom Setup 界面如图 2.20 所示。

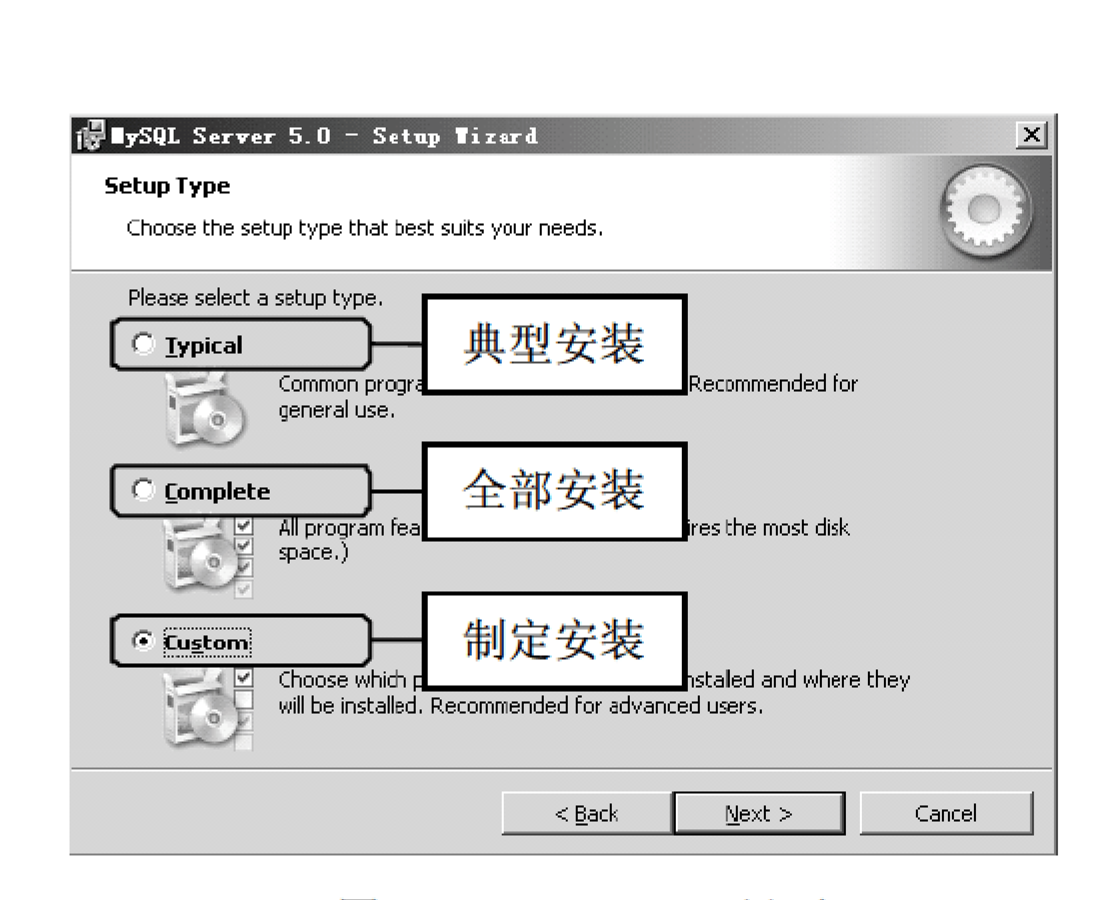




图 2.19 Setup Type 界面

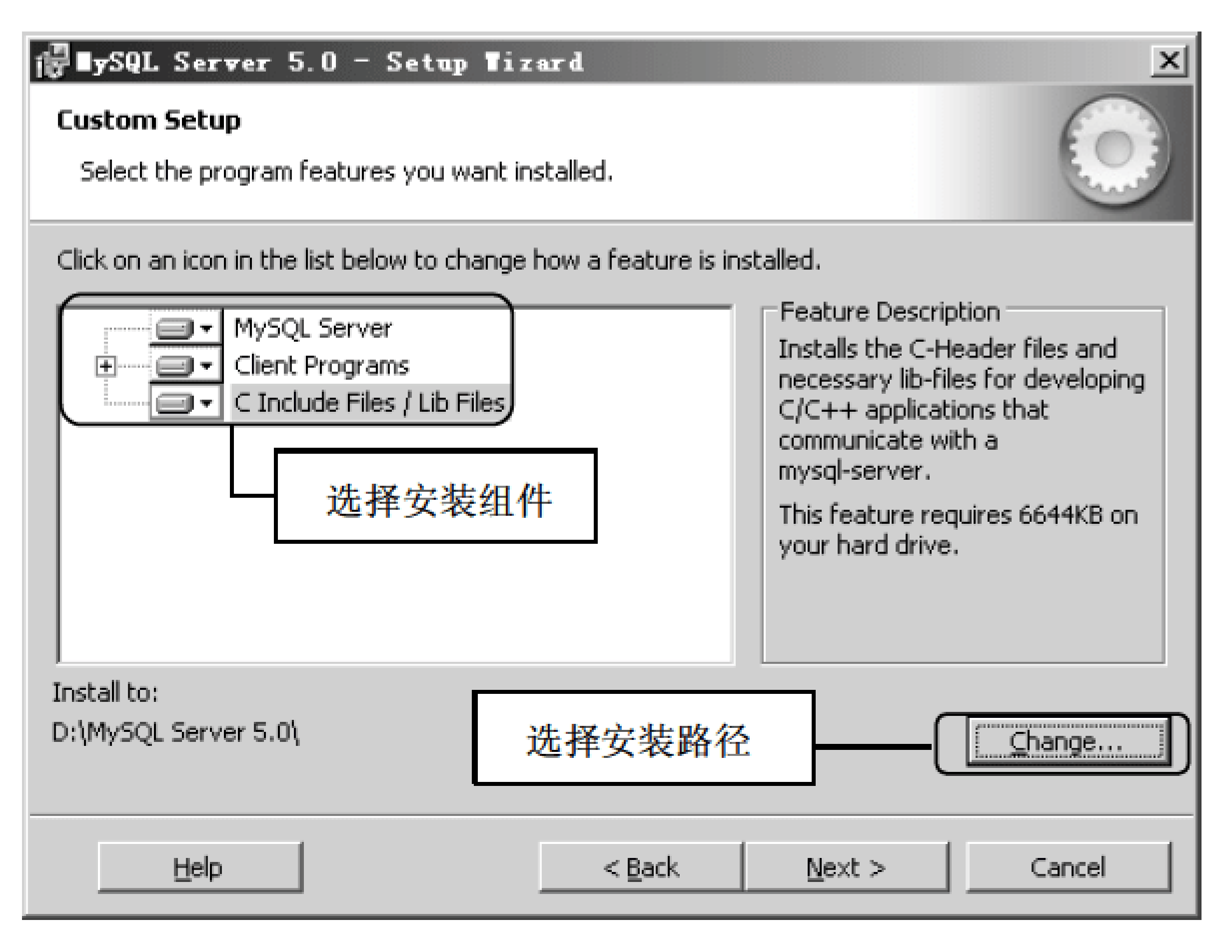


图 2.20 Custom Setup 界面

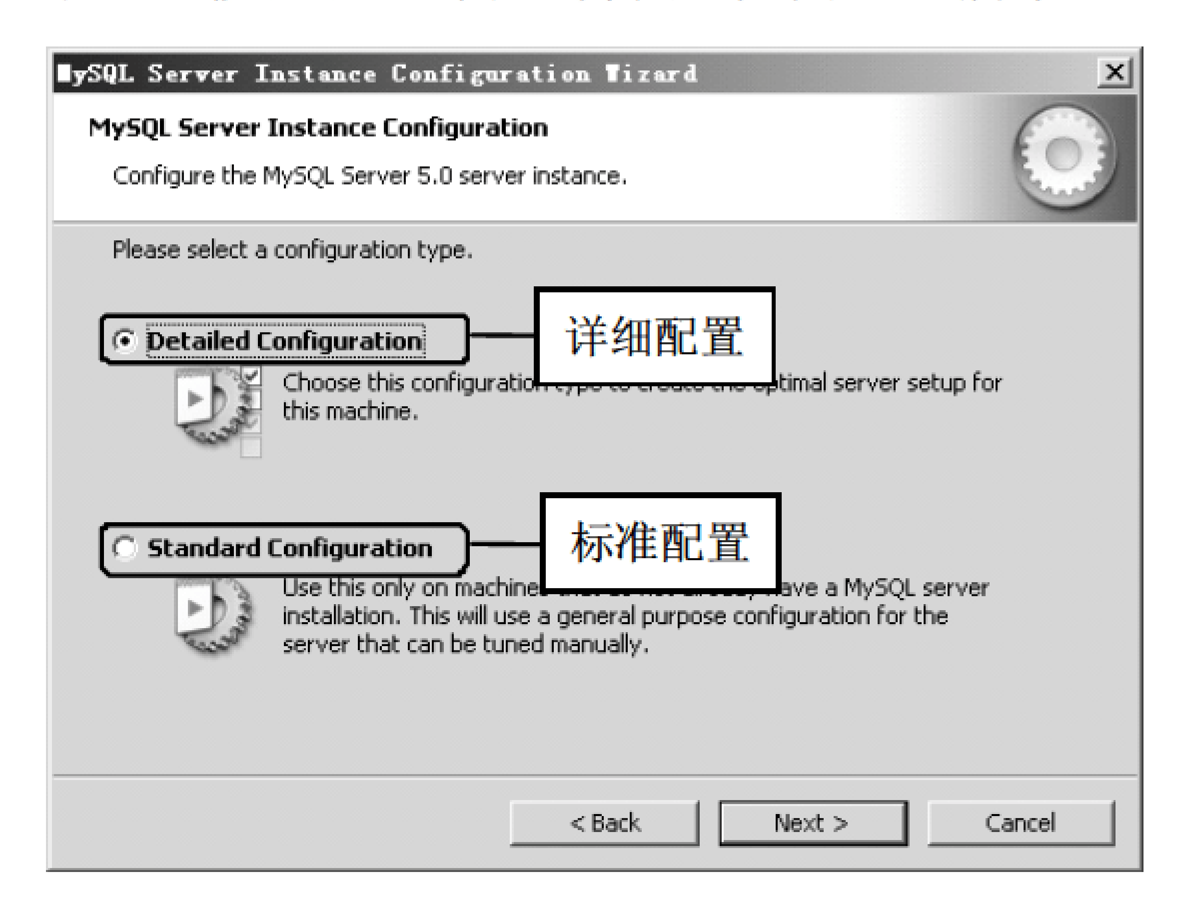
（4）该页面是准备安装界面，界面中显示了用户所选择的安装类型、路径等信息。如果发现前面的选项有误，可以单击 Back 按钮重新选择；如果正确，则单击 Install 按钮开始安装文件。

（5）文件安装完成后，会出现一些关于 MySQL 的功能和版本的介绍。连续单击 Next 按钮，将会进入 MySQL 服务器配置界面（MySQL Server Instance Configuration Wizard）。配置界面如图 2.21 所示。

（6）界面有两个选项：“详细配置”（默认）和“标准配置”。这里选择“详细配置”选项，单



击 Next 按钮进入到下一界面，如图 2.22 所示。



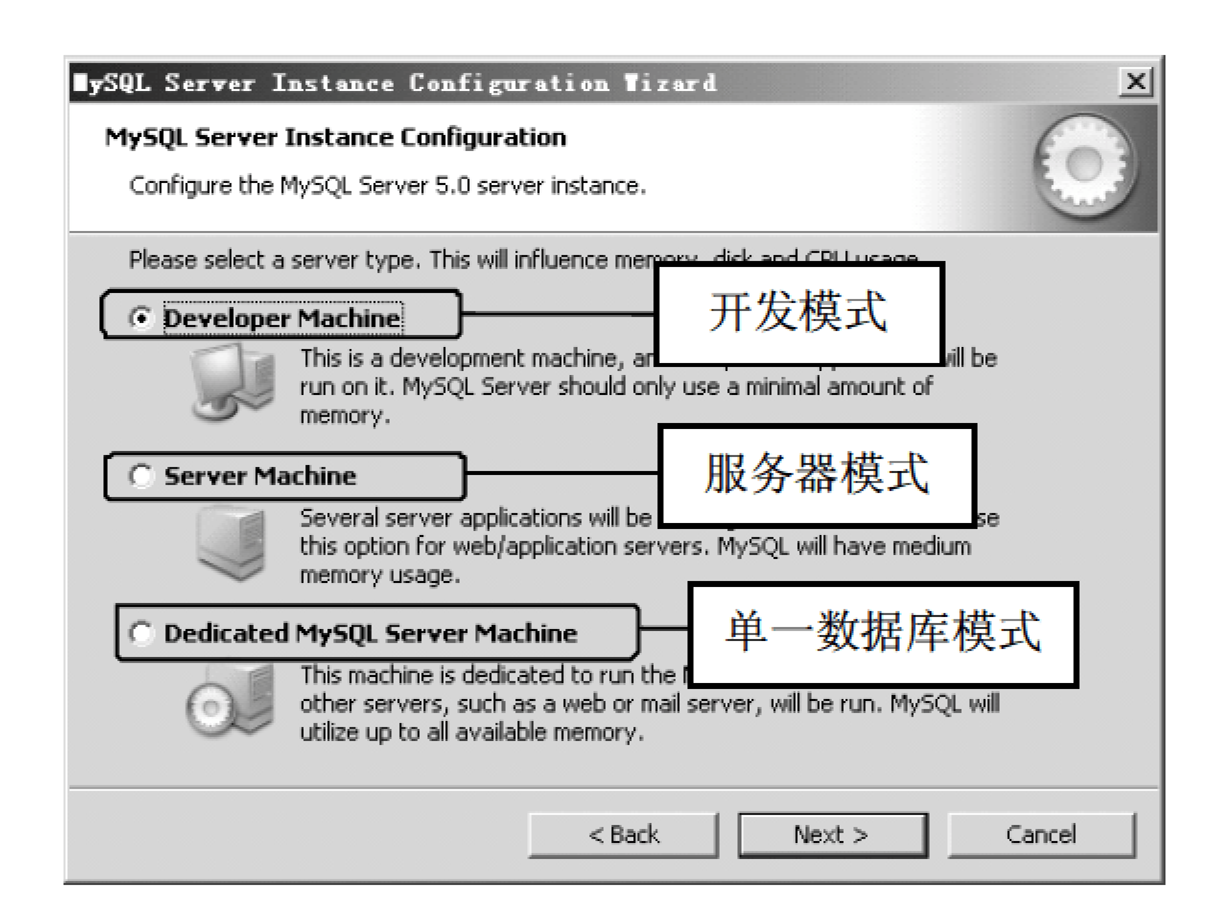
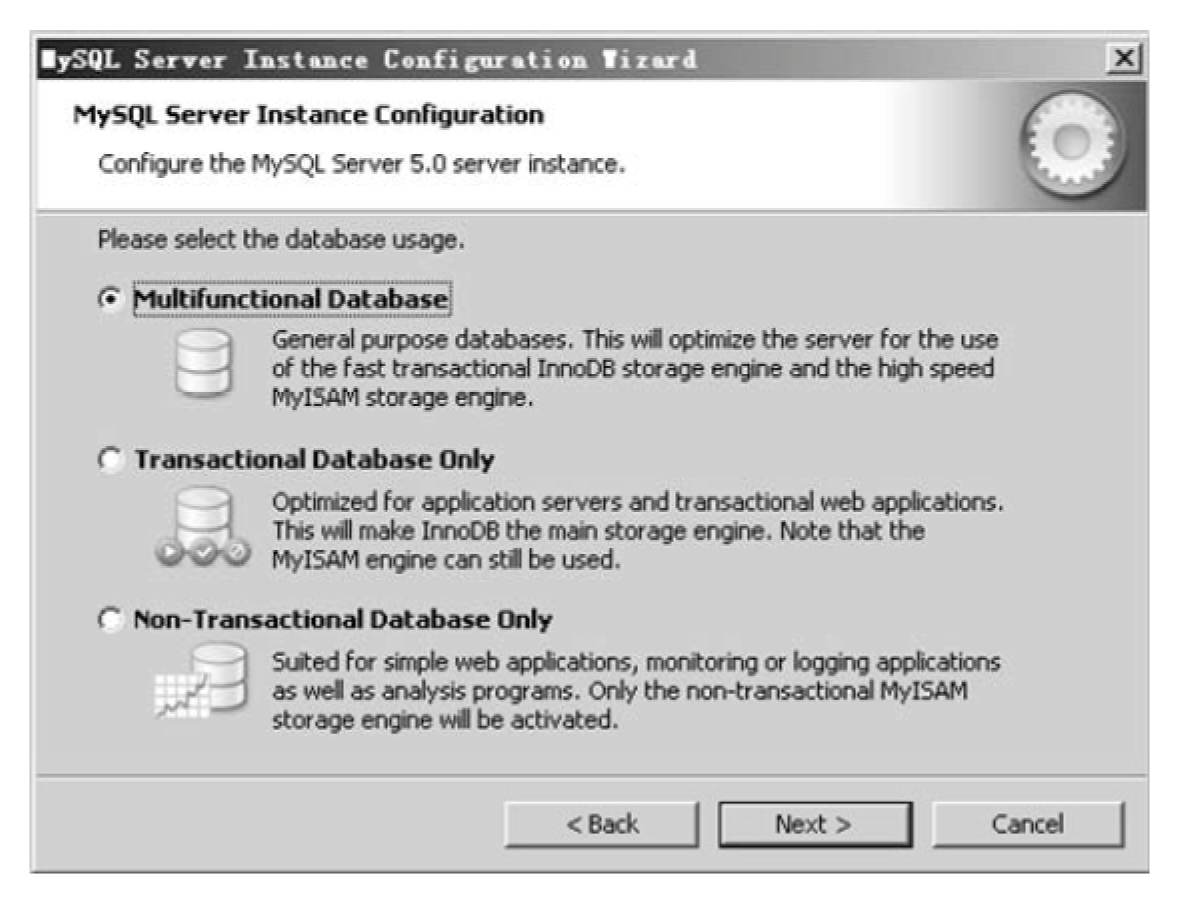


图 2.21　MySQL 服务器配置类型　　　　图 2.22　服务器运行模式

（7）选择第一个默认项即可（即开发模式，MySQL 服务器占用最小的内存空间。作为本地测试使用完全足够）。选择完毕后，单击 Next 按钮进入下一项配置。

（8）选择数据库的类型。可支持 MyISAM、InnoDB 等多种类型库的数据系统，也可只支持其中一种类型库。这里选择默认的第一项 Multifunctional Database，即支持多种类型库。选择完毕后的界面效果如图 2.23 所示。

（9）单击 Next 按钮进入下一个界面。本界面为 InnoDB 数据文件选择路径，这里选择 D 盘下的 MySQL Datafiles 目录。选取分区时要注意选取分区的剩余空间大小。选择后的界面如图 2.24 所示。



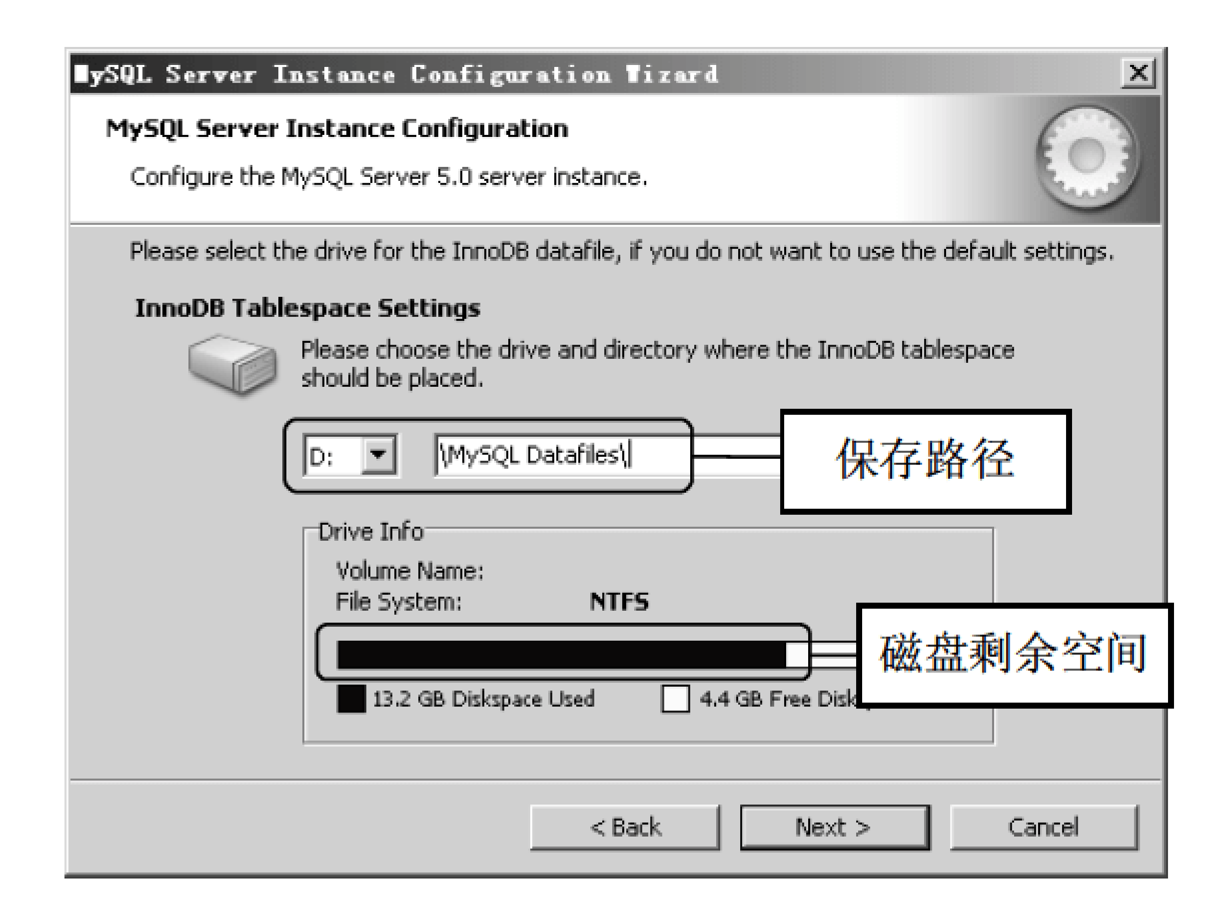


图 2.23　选择数据库类型　　　　图 2.24　为 InnoDB 设置路径

（10）单击 Next 按钮进入下一界面。设置同时连接服务器的最大值，这里可以选择默认的第一项，或者选择第三项自定义连接。第二项的最大连接数为 500。选择后的界面如图 2.25 所示。

（11）单击 Next 按钮进入下一界面，该界面为 MySQL 服务器的端口设置，默认 3306 即可。设置完毕后单击 Next 按钮。

（12）选择 MySQL 的默认字符集，这里选择 GB2312。单击 Next 按钮进入下一界面。

（13）选择 MySQL 服务器是否自动运行。如果要在 Windows 环境变量 PATH 中加入 MySQL 执行路径，那么将下面的“Include Bin Directory in Windows PATH”复选框选中。运行界面如图 2.26 所示。

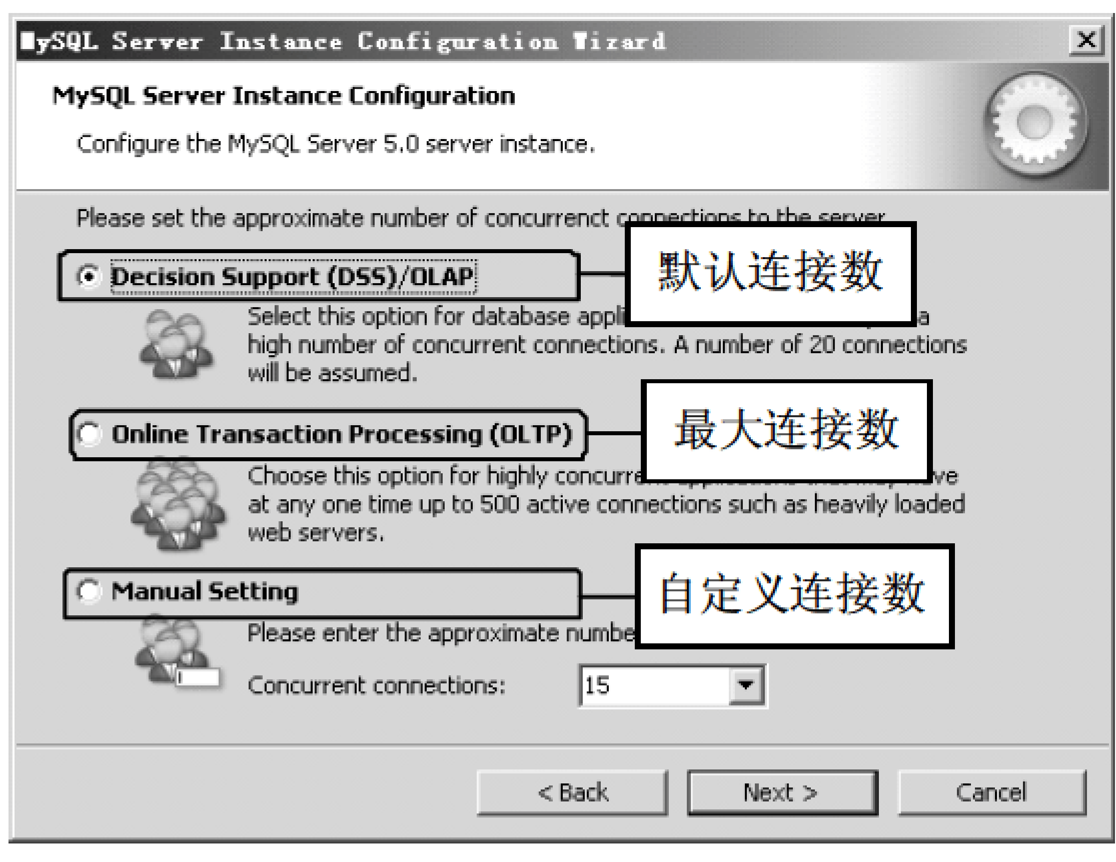


图 2.25　选择同时连接服务器的最大值

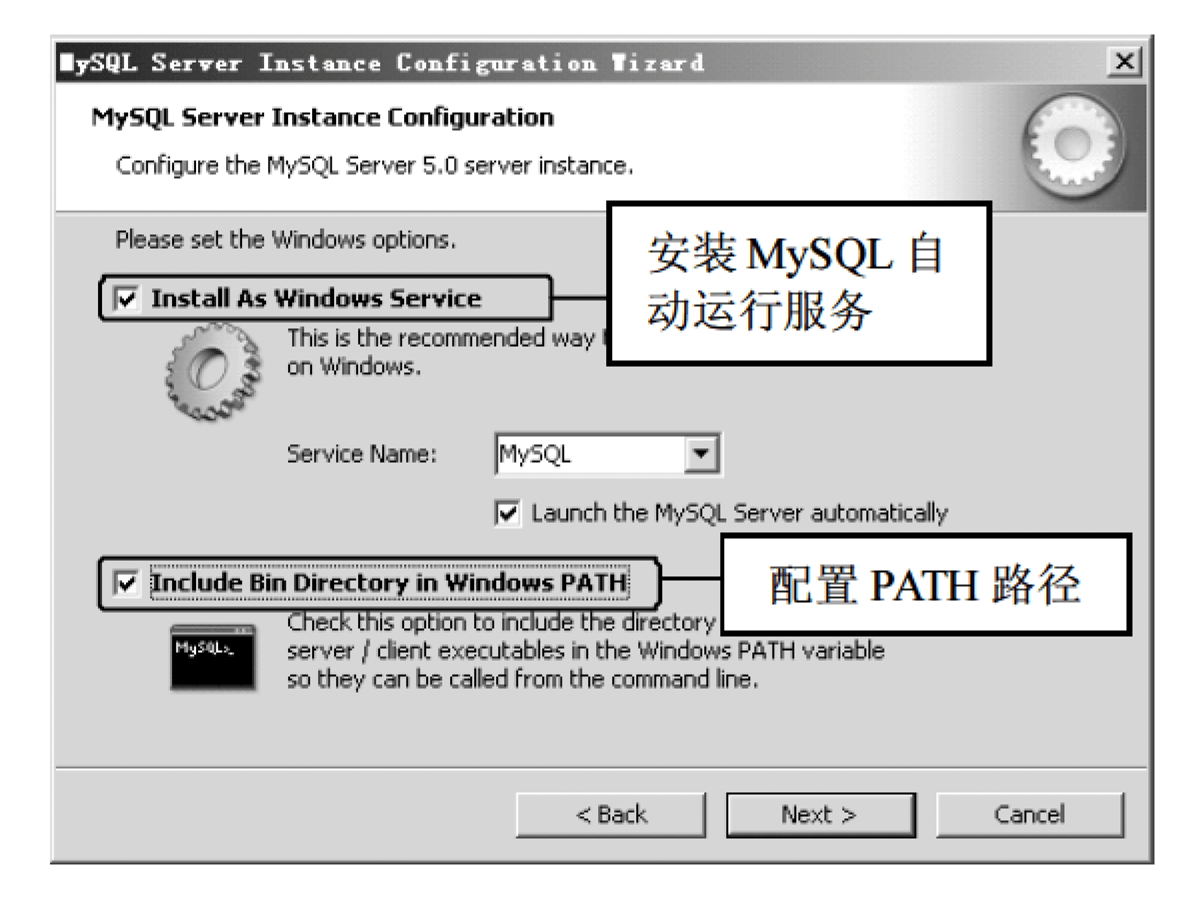


图 2.26　选择 MySQL 服务器的启动方式

（14）单击 Next 按钮，进入到权限设置界面。这里可以设置用户登录密码。因为本书中所有涉及数据库实例的密码都为 root，所以这里建议用户也设置为 root（Enable root access from remote machines）。复选框用于设置是否允许 root 用户远程登录数据库。如果选中 Create An Anonymous Account 复选框，则创建一个允许任何人访问数据库的账号，这里不建议选中。运行界面如图 2.27 所示。

（15）单击 Next 按钮，打开准备执行界面。如果配置没有问题，单击 Execute 按钮开始执行操作。界面如图 2.28 所示。

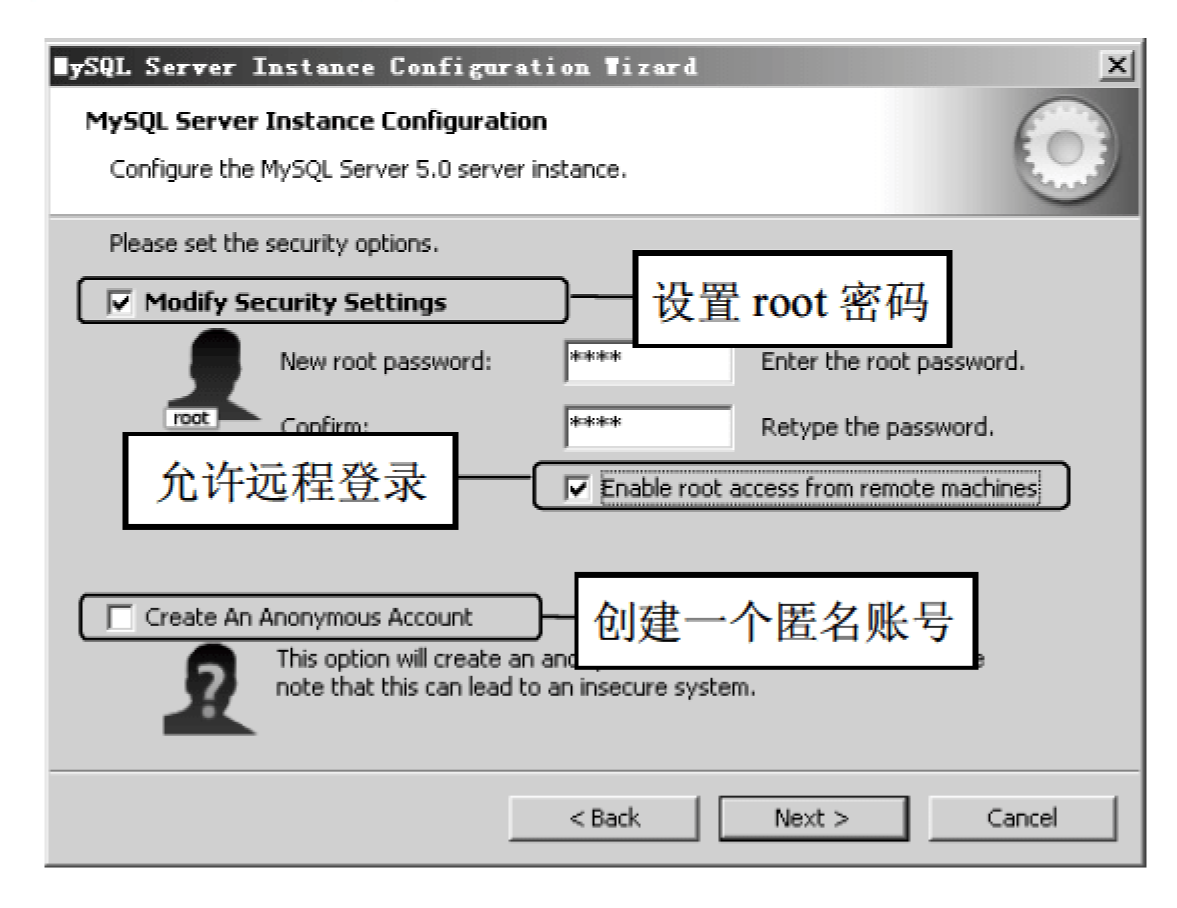


图 2.27　账号配置界面

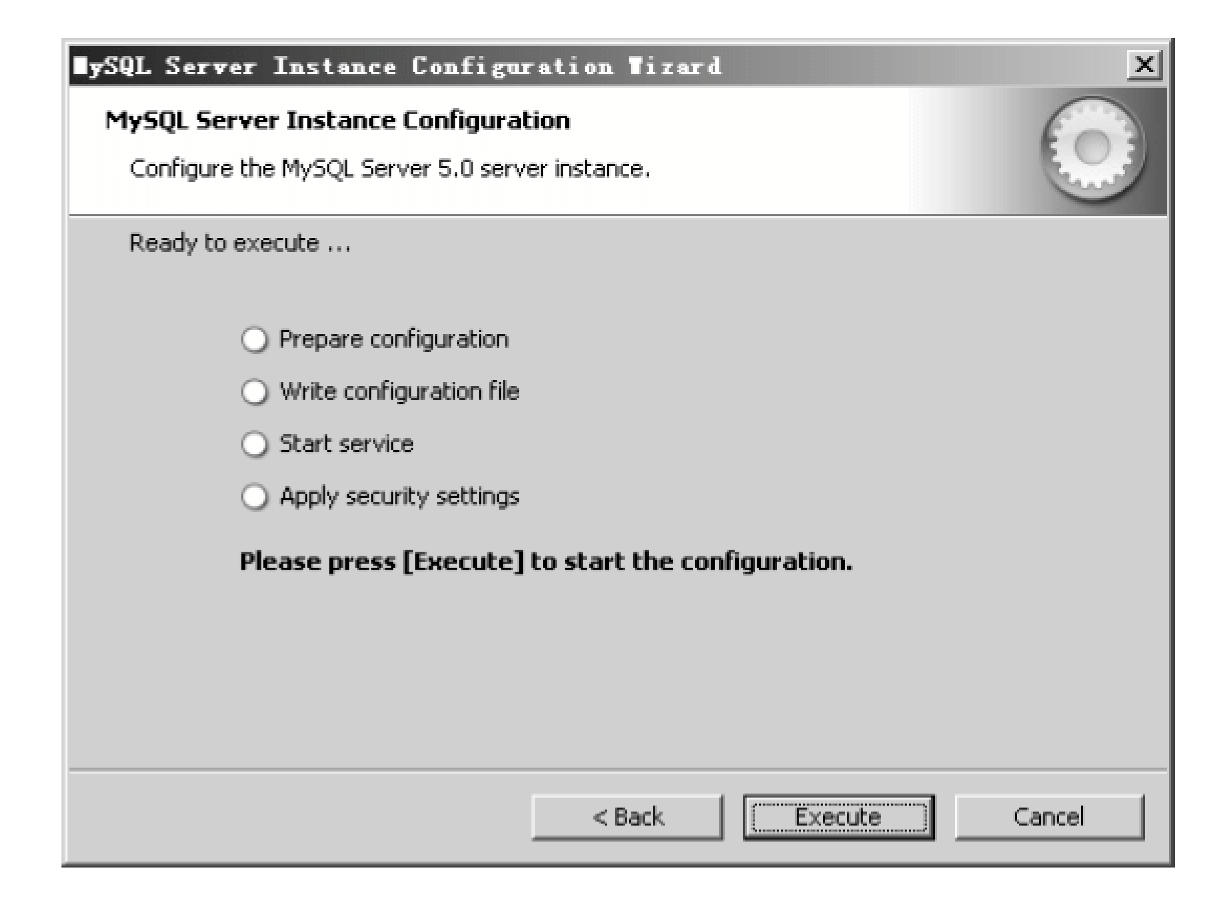


图 2.28　准备执行界面

（16）安装完成后，单击 Finish 按钮结束。



2.3　在 Linux 下的安装配置

视频讲解：光盘\TM\lx\2\在 Linux 下的安装配置.exe

在 Linux 下搭建 PHP 环境比 Windows 下要复杂得多，除了 Apache、PHP 等软件外，还要安装一些相关工具，并设置必要参数。而且，如果要使用 PHP 扩展库，还要进行编译，如本书中使用到的 SOAP、MHASH 等扩展库。

安装之前要准备的安装包如下：

httpd-2.2.8.tar.gz
php-5.2.5.tar.gz
mysql-5.0.51a-Linux-i686.tar.gz
libxml2-2.6.26.tar.gz

2.3.1　安装 Apache 服务器

安装 Apache 服务器，首先需要打开 Linux 终端（Linux 下几乎所有的软件都需要在终端下安装）。选择 RedHat9 的“主菜单”/“系统工具”命令，在弹出的子菜单中选择“终端”命令。下面介绍安装 Apache 的具体步骤。

（1）进入 Apache 安装文件的目录下，如/usr/local/work。

```
cd /usr/local/work/
```

（2）解压安装包。解压完成后，进入 httpd2.2.8 目录中。

```
tar xfz httpd2.2.8.tar.gz
cd httd2.2.8
```

（3）建立 makefile，将 Apache 服务器安装到 usr/local/Apache2 下。

```
./configure –prefix=/usr/local/Apache2 –enable-module=so
```

（4）编译文件。

```
make
```

（5）开始安装。

```
make install
```

（6）安装完成后，将 Apache 服务器添加到系统启动项中，最后重启服务器。

```
/usr/local/Apache2/bin/Apachectl start >> /etc/rc.d/rc.local
/usr/local/Apache2/bin/Apachectl restart
```

（7）打开 Mozilla 浏览器，在地址栏中输入“http://localhost/”，按 Enter 键后如果看到如图 2.29 所示的页面，说明安装 Apache 服务器成功。

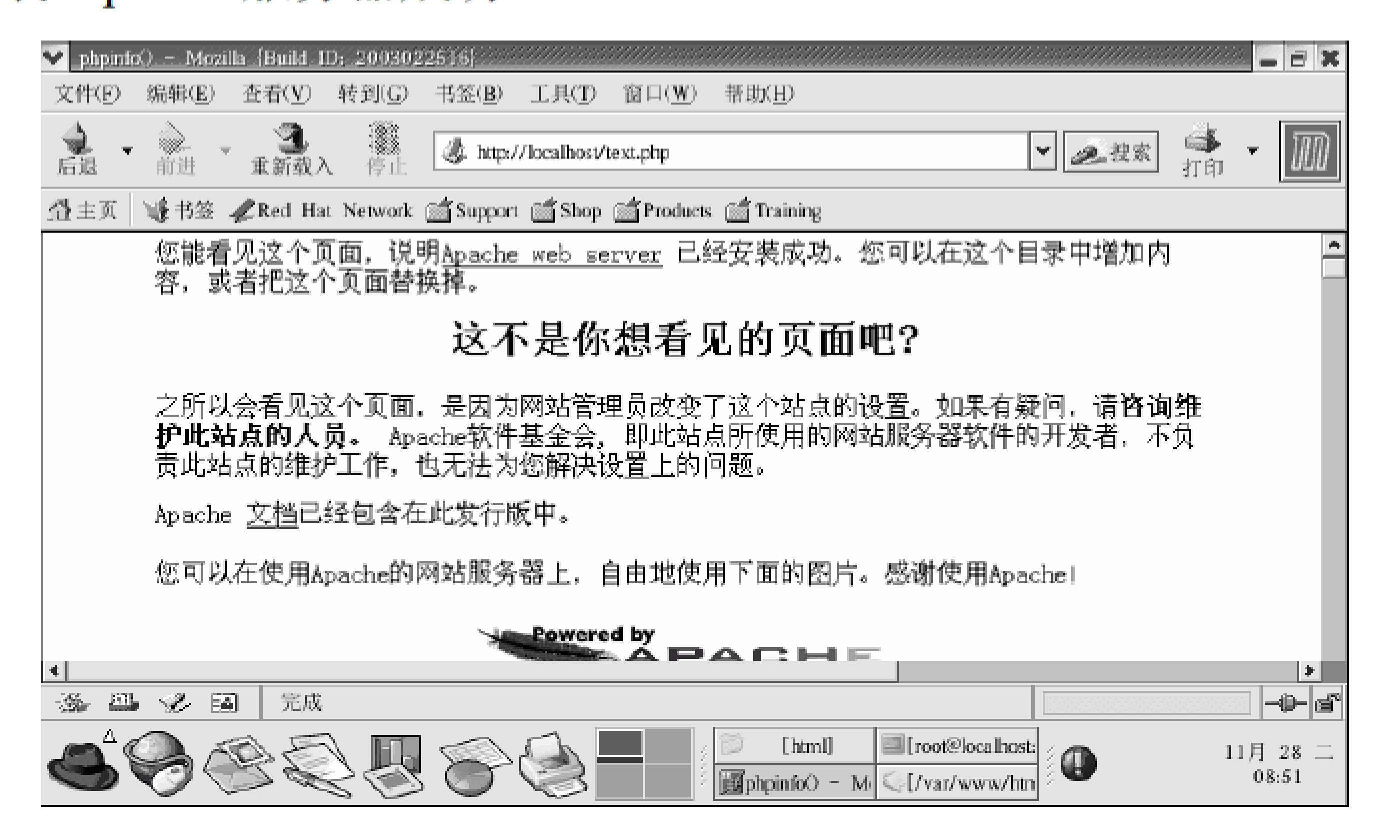


图 2.29　Linux 下的 Apache 服务器安装

2.3.2　安装 MySQL 数据库

安装 MySQL 比 Apache 稍复杂一些，因为需要创建 MySQL 账号，并将新建账号加入到组群。安装步骤如下：

（1）创建 MySQL 账号，并加入组群。

```
groupadd mysql
useradd –g mysql mysql
```

（2）进入 MySQL 的安装目录，将其解压（如目录为/usr/local/mysql）。

```
cd /usr/local/mysql
tar xfz /usr/local/work/mysql-5.0.51a-Linux-i686.tar.gz
```

（3）考虑到 MySQL 数据库升级的需要，通常以链接的方式建立/usr/local/mysql 目录。

```
ln –s mysql-5.0.51a-Linux-i686.tar.gz mysql
```

（4）进入 MySQL 目录，在/usr/local/mysql/data 中建立 MySQL 数据库。

```
cd mysql
scripts/mysql_install_db –user=mysql
```

（5）修改文件权限。

```
chown –R root
chown –R mysql data
chgrp –R mysql
```

（6）至此，MySQL 安装成功。用户可以通过在终端中输入命令启动 MySQL 服务。



```
/usr/local/mysql/bin/mysqld_safe –user=mysql &
```

启动后输入命令，进入 MySQL。

```
/user/local/mysql/bin/mysql -uroot
```

2.3.3　安装 PHP 5 语言

安装 PHP 5 之前，首先需要查看 libxml 的版本号。如果 libxml 版本号小于 2.5.10，则需要先安装 libxml 高版本。安装 libxml 和 PHP 5 的步骤如下（如果不需要安装 libxml，直接执行 PHP 5 的安装步骤即可）：

（1）将 libxml 和 PHP 5 复制到/usr/local/work 目录下，并进入该目录。

```
mv php-5.2.5.tar.gz libxml2-2.6.26.tar.gz /usr/local/work
cd /usr/local/work
```

（2）分别将 libxml2 和 PHP 解压。

```
tar xfz libxml2-2.6.62.tar.gz
tar xfz PHP-5.2.5.tar.gz
```

（3）进入 libxml2 目录，建立 makefile，将 libxml 安装到/usr/local/libxml2 下。

```
cd libxml2-2.6.62
./configure –prefix=/usr/local/libxml2
```

（4）编译文件。

```
makefile
```

（5）开始安装。

```
make install
```

（6）libxml2 安装完毕后，开始安装 PHP 5。进入 php-5.2.5 目录下。

```
cd ../php-5.2.5
```

（7）建立 makefile。

```
./configure –with-apxs2=/usr/local/Apache2/bin/apxs
--with-mysql=/usr/local/mysql
--with-libxml-dir=/usr/local/libxml2
```

（8）开始编译。

```
make
```

（9）开始安装。

```
make install
```

（10）复制 php.ini-dist 或 php.ini-recommended 到/usr/local/lib 目录，并命名为 php.ini。

```
cp php.ini-dist /usr/local/lib/php.ini
```

（11）更改 httpd.conf 文件相关设置，该文件位于/usr/local/Apache2/conf 中。找到该文件中的如下指令行：

```
AddType application/x-gzip .gz .tgz
```

在该指令后加入如下指令：

```
AddType application/x-httpd-php .php
```

重新启动 Apache，并在 Apache 主目录下建立文件 phpinfo.php。

```
<?php
phpinfo();
?>
```

在 Mozilla 浏览器中输入“http://localhost/phpinfo.php”，按 Enter 键，如果出现如图 2.30 所示的页面，则 PHP 安装成功。

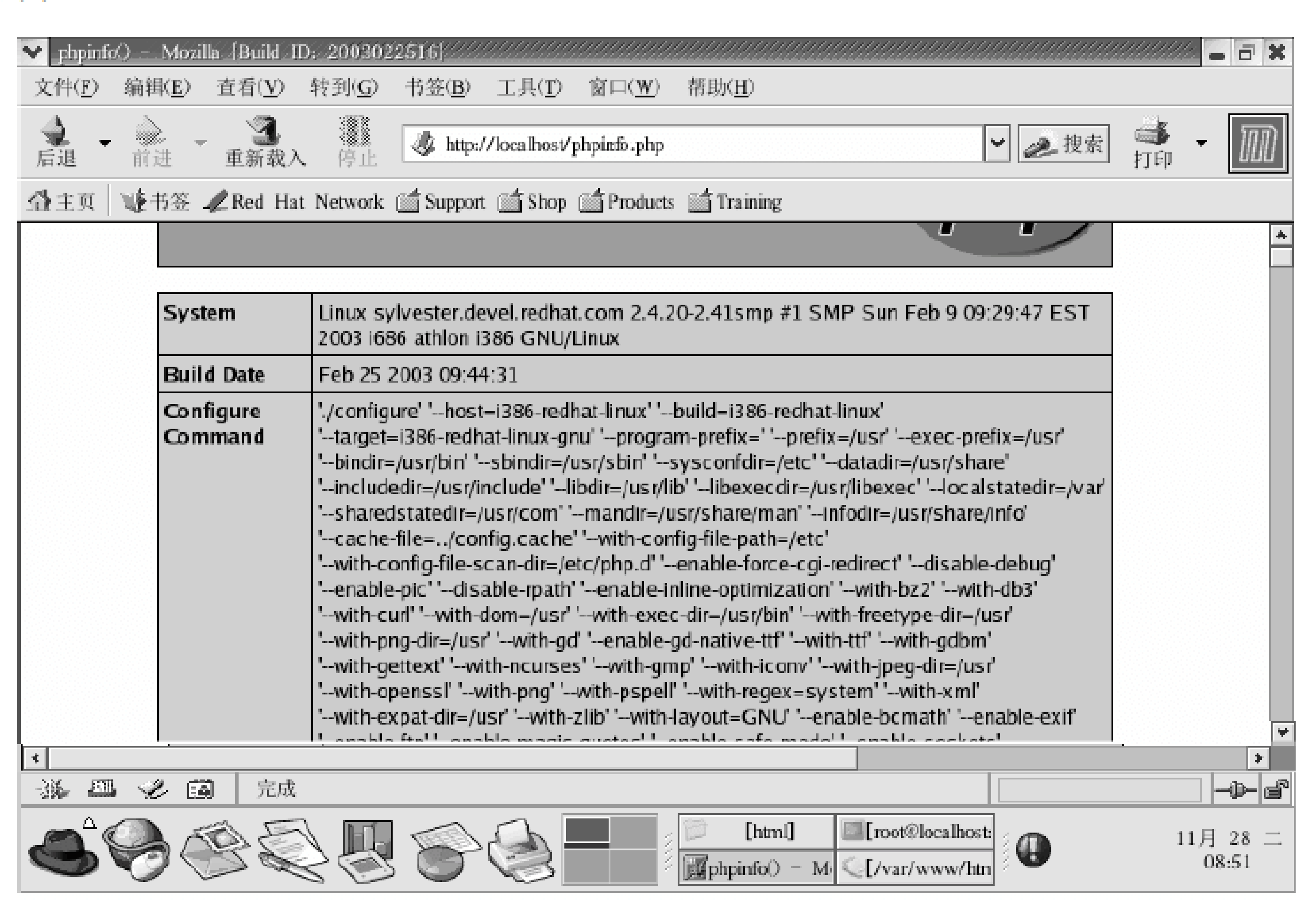
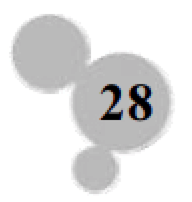


图 2.30　phpinfo 信息

2.4　PHP 常用开发工具

“工欲善其事，必先利其器”。随着 PHP 的发展，大量优秀的开发工具纷纷出现。找到一个适合自己的开发工具，不仅可以加快学习进度，而且能够在以后的开发过程中及时发现问题，少走弯路。下面将介绍两款目前流行的开发工具。



2.4.1　Zend Studio

视频讲解：光盘\TM\lx\2\Zend Studio 开发工具.exe

Zend Studio 是目前公认最好的 PHP 开发工具，它包括编辑、调试、配置 PHP 程序所需要的客户及服务器组件，尤其是功能齐全的调试功能，让 PHP 错误不再棘手。

Zend Studio 的缺点主要是速度慢，安装繁琐，而且需要收费，但可以下载试用版。下载地址为 http://www.zend.com/store/products/zend-studio.php。

Zend Studio for Eclipse 的安装步骤如下：

（1）运行安装文件，进行安装准备工作，如图 2.31 所示。

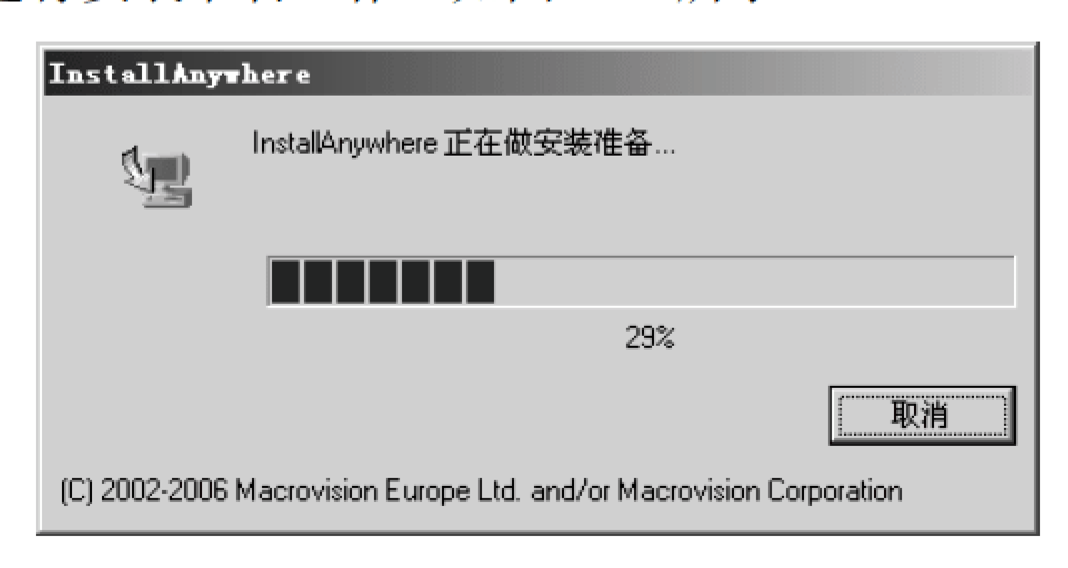
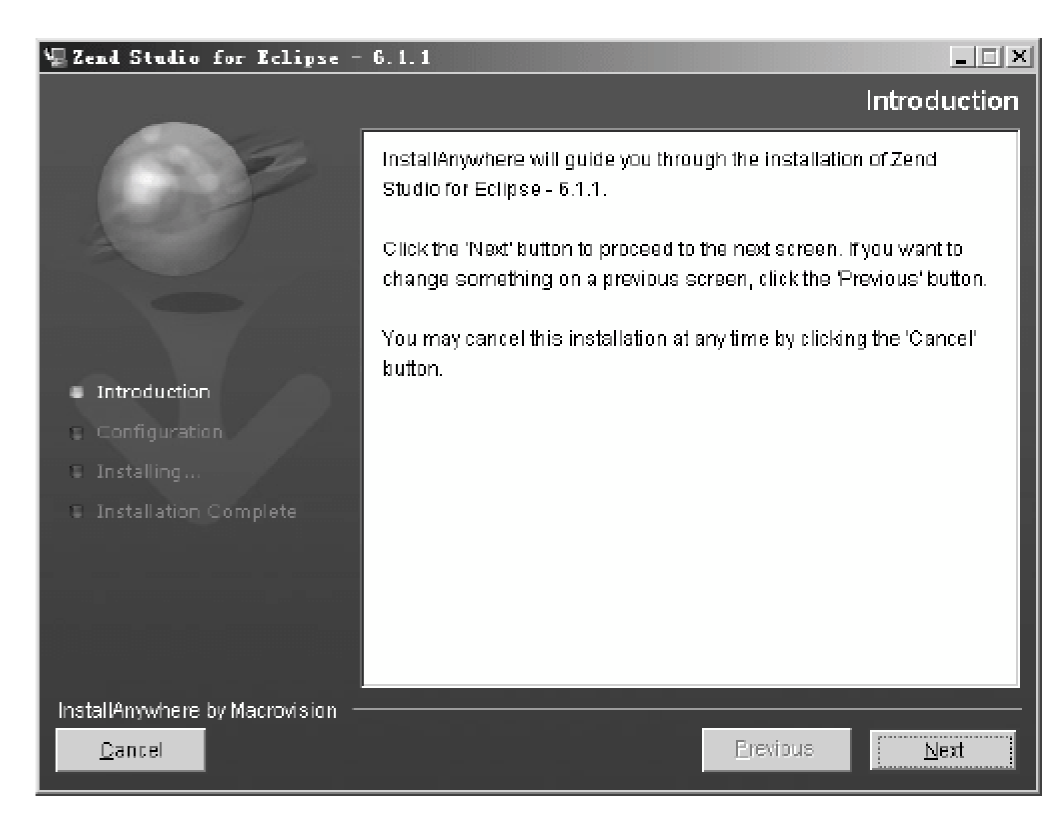


图 2.31　安装 Zend Studio for Eclipse 的准备工作

（2）进入到安装页面，如图 2.32 所示，单击 Next 按钮。

（3）阅读服务器条款，选中 I accept the terms of the License Agreement 单选按钮，如图 2.33 所示。单击 Next 按钮进入下一页面。



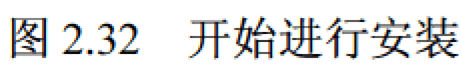

图 2.32　开始进行安装



图 2.33　阅读服务条款

（4）选择安装内容，将 Zend 浏览器和 Zend Studio 全部选中，如图 2.34 所示。单击 Next 按钮进入到下一页面。

（5）设置文件的安装目录，用户可以任意指定安装位置，确定后单击 Next 按钮进行下一项操作，如图 2.35 所示。

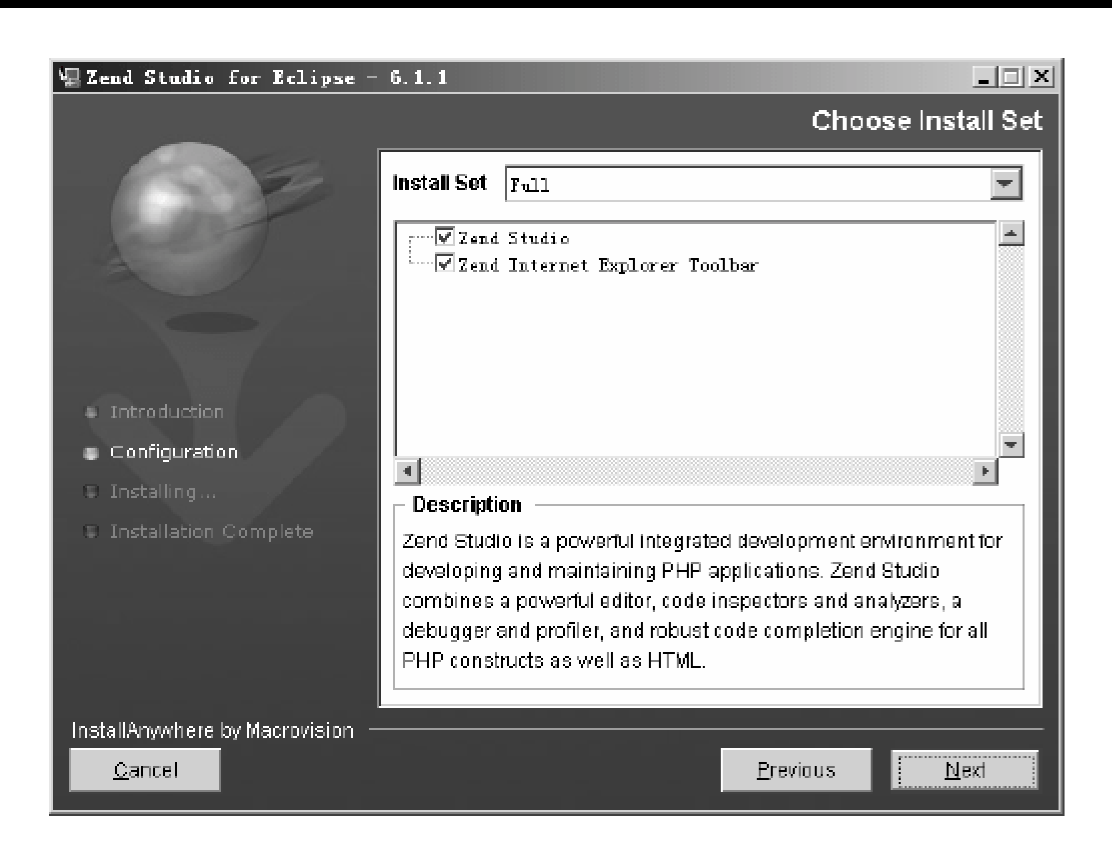


图 2.34　选择安装内容

图 2.35　设置工具安装位置

（6）定义 Zend 支持的文件格式，全部选中，如图 2.36 所示。

（7）确认安装信息，如图 2.37 所示。单击 Install 按钮开始进行安装。

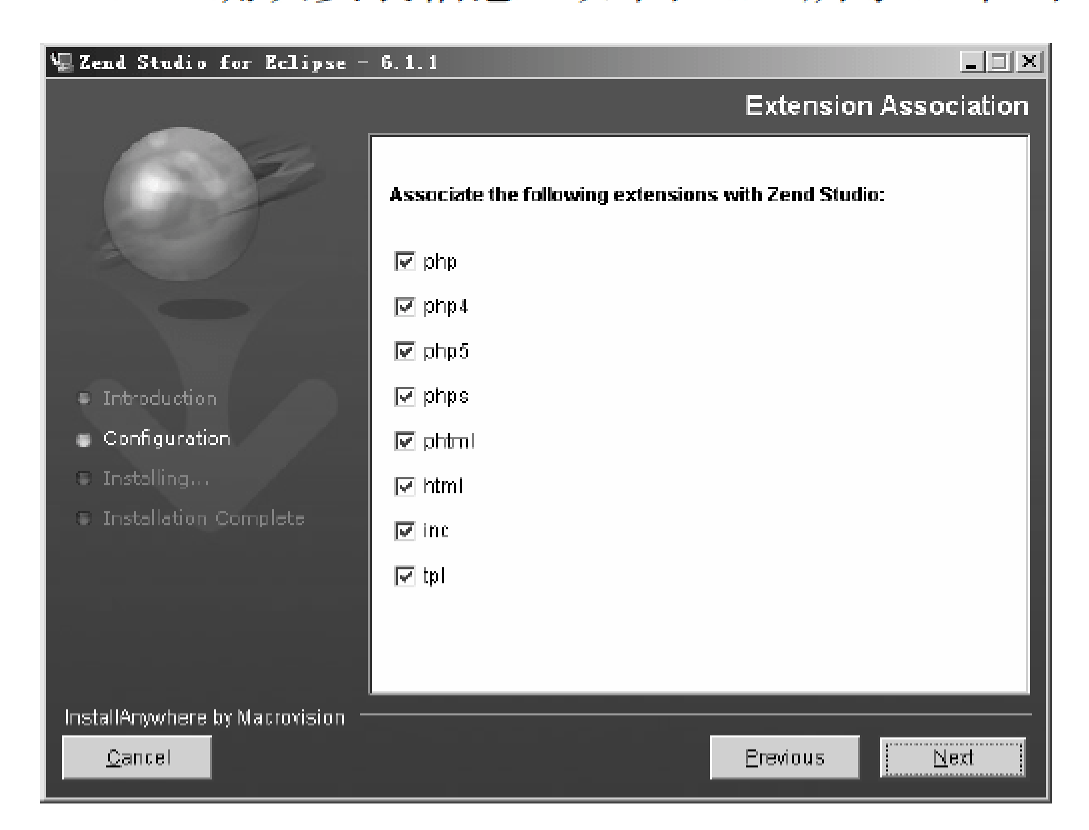


图 2.36　定义 Zend 支持的文件格式



图 2.37　确认安装信息

（8）安装进行中，如图 2.38 所示。

（9）安装完成，单击 Done 按钮，如图 2.39 所示。



图 2.38　Zend 安装进行中

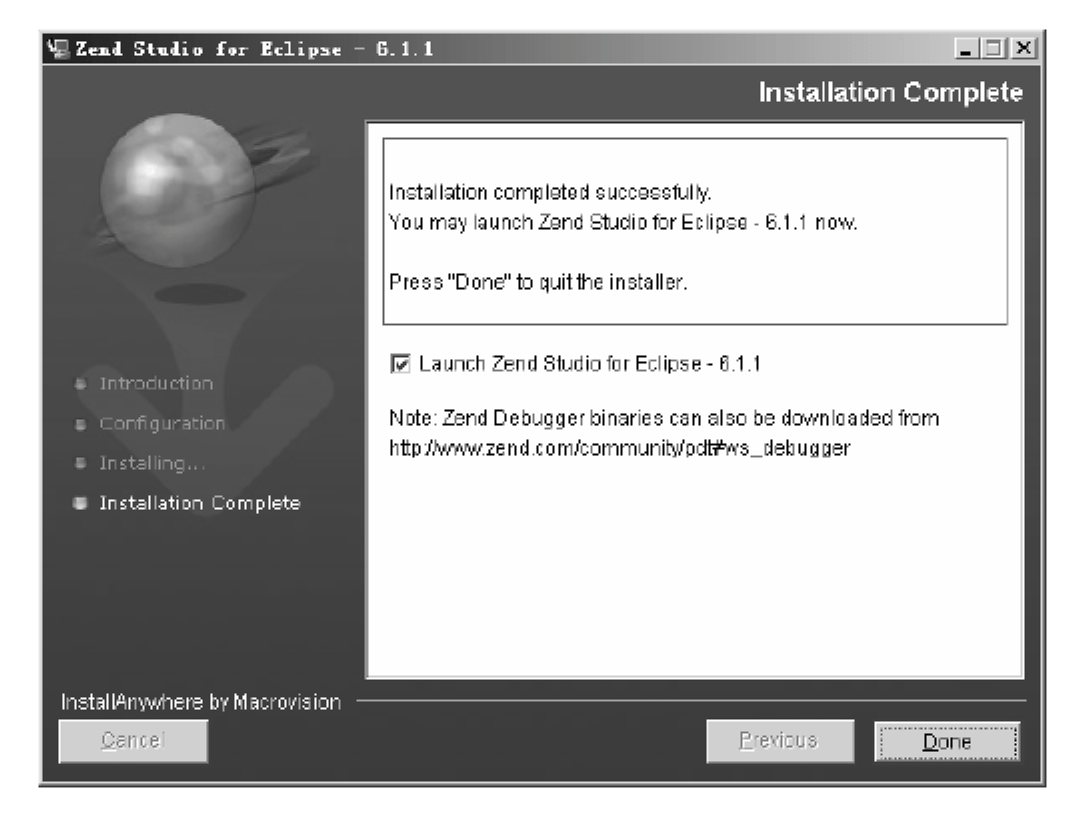
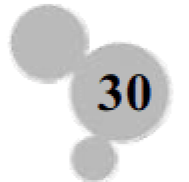


图 2.39　安装成功



Zend Studio for Eclipse 安装成功，下面介绍如何使用这个工具。

（1）运行 Zend Studio，进入 Zend Studio 的欢迎页面，如图 2.40 所示。

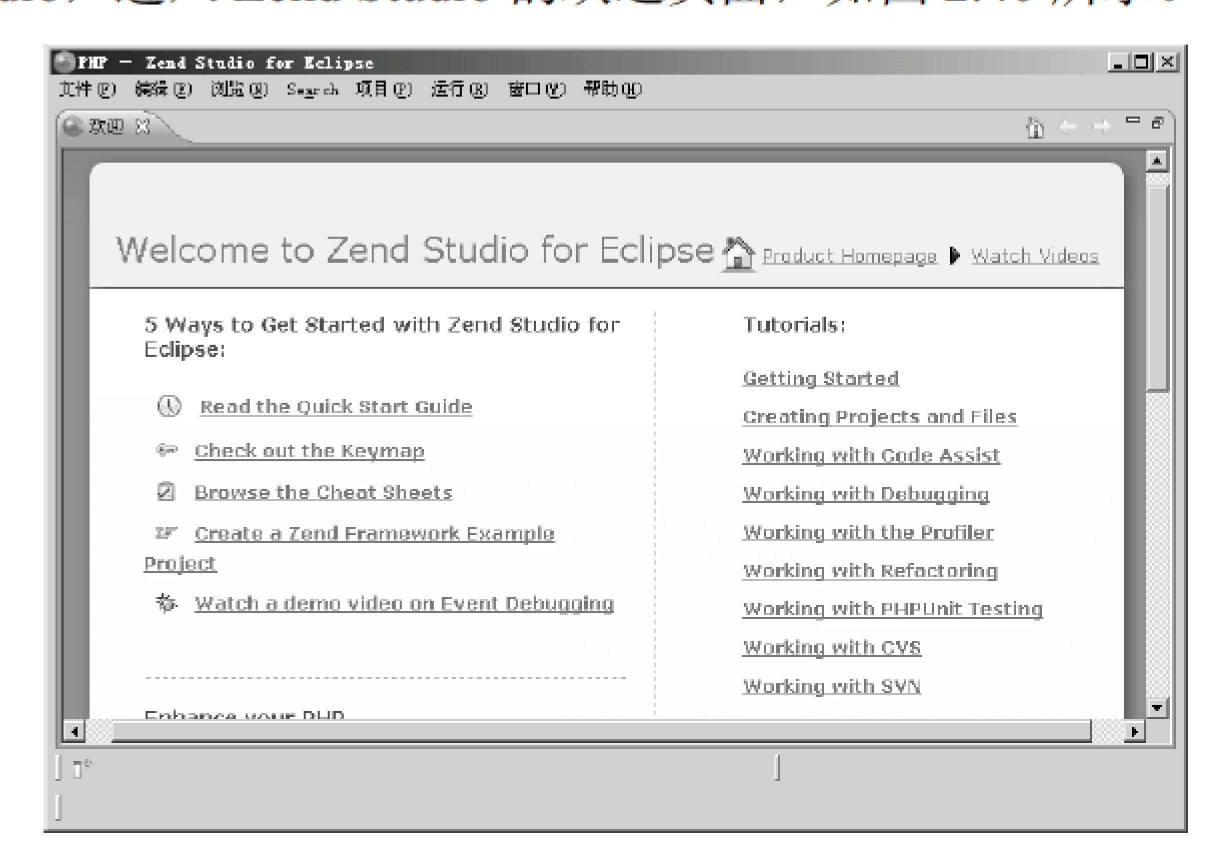


图 2.40　Zend Studio 的欢迎页面

（2）选择“文件”/“切换工作空间”/“其他”命令，设置工作空间，将项目保存在指定名称的文件夹下，如图 2.41 所示。本实例中将 F:\AppServ\www\mr 文件夹作为工作空间。

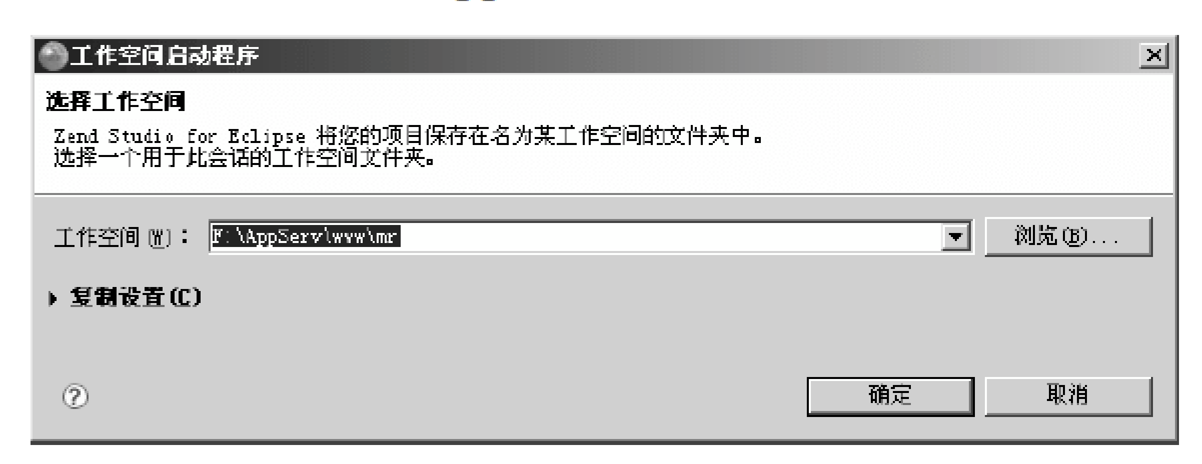


图 2.41　选择工作空间

（3）进入 Zend Studio 的工作台，如图 2.42 所示。

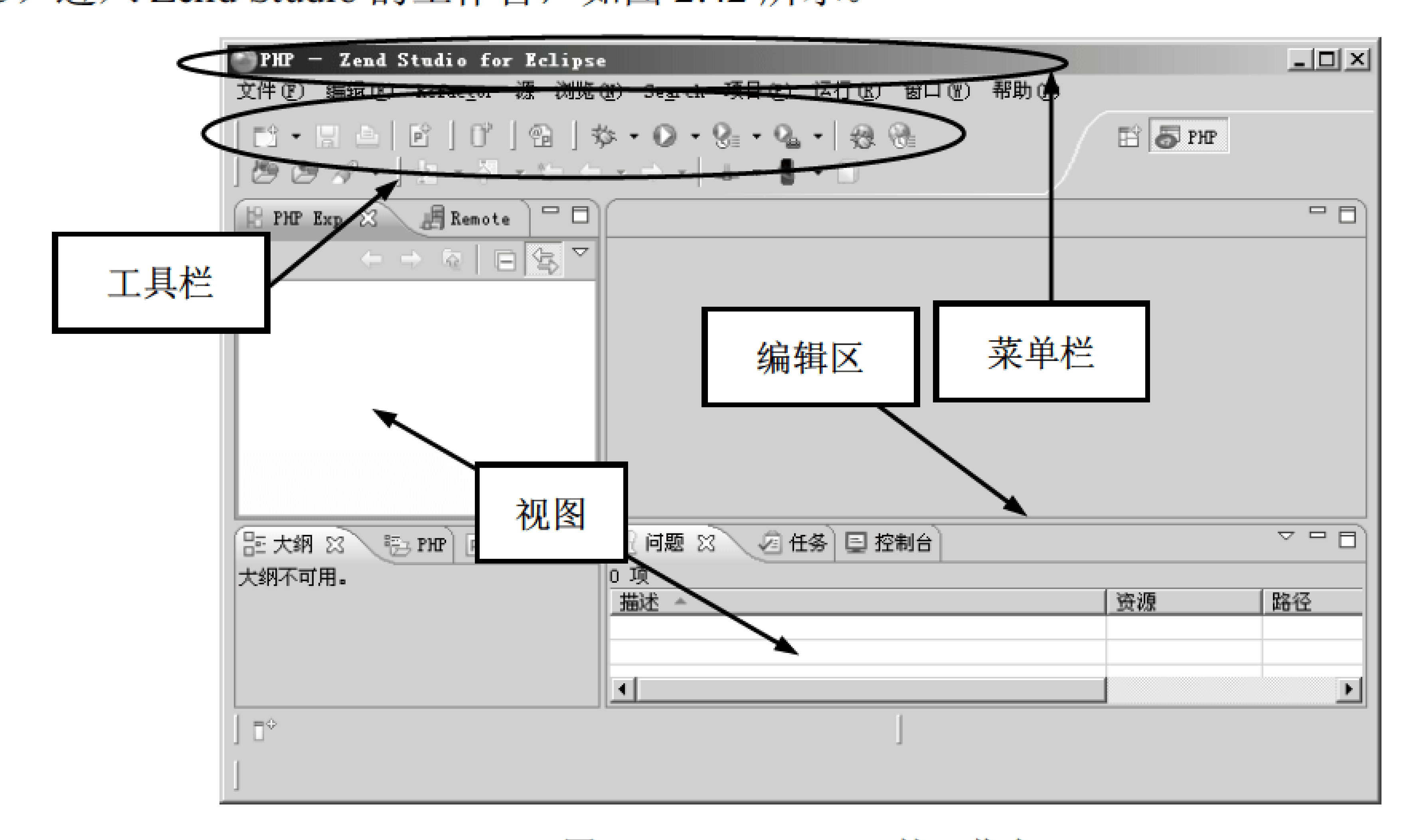


图 2.42　Zend Studio 的工作台

（4）下面创建一个 PHP 项目。选择“文件”/“新建”/PHP Project 命令，创建一个 PHP 项目，如图 2.43 所示。

（5）进入新项目创建窗口中，在该窗口中设置项目名称 01；指定项目文件存储位置，使用默认值即可；选择 PHP 的版本，设置是否支持 JavaScript 脚本，如图 2.44 所示。

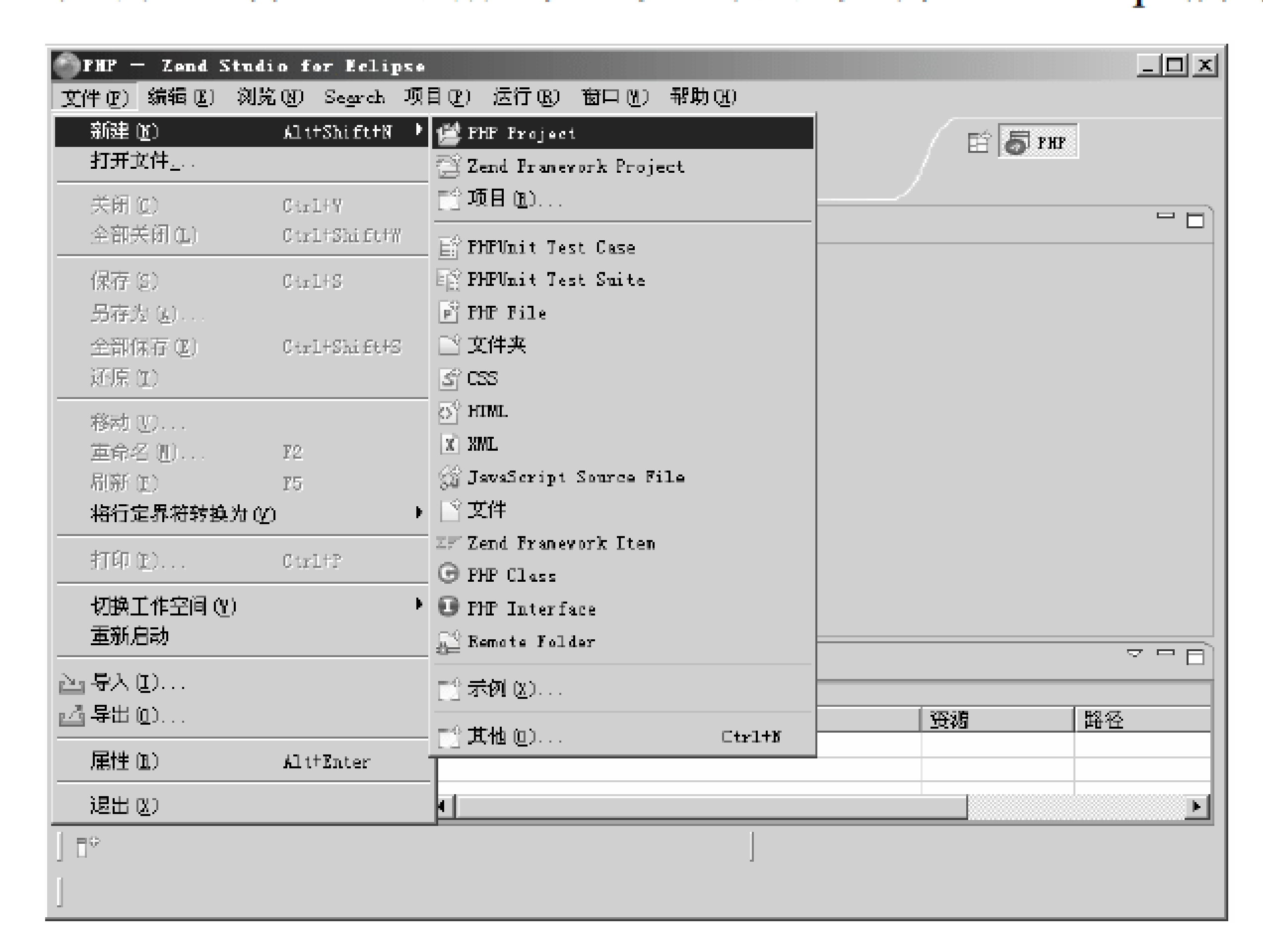


图 2.43　创建 PHP 项目

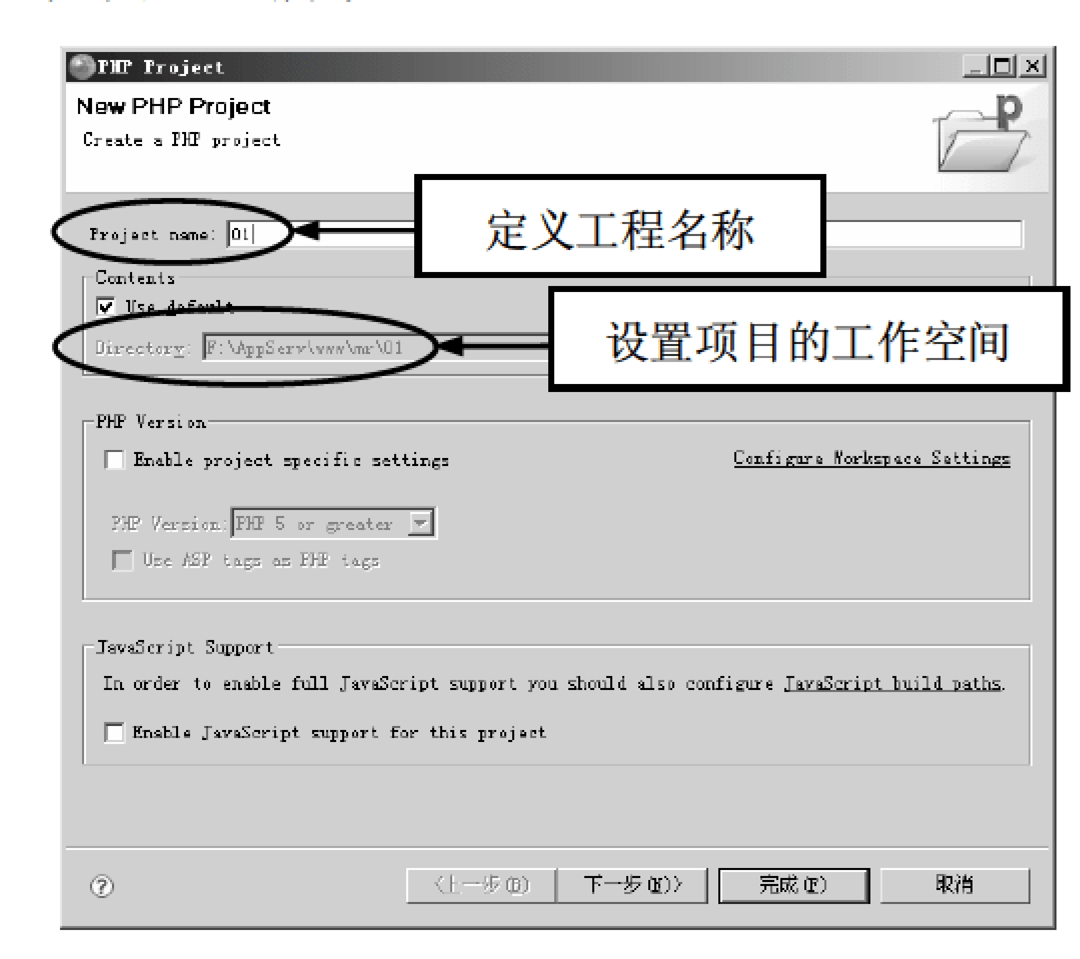


图 2.44　新建工程

（6）接下来在项目 01 中创建 PHP 文件，编写一个简单实例，如图 2.45 所示。

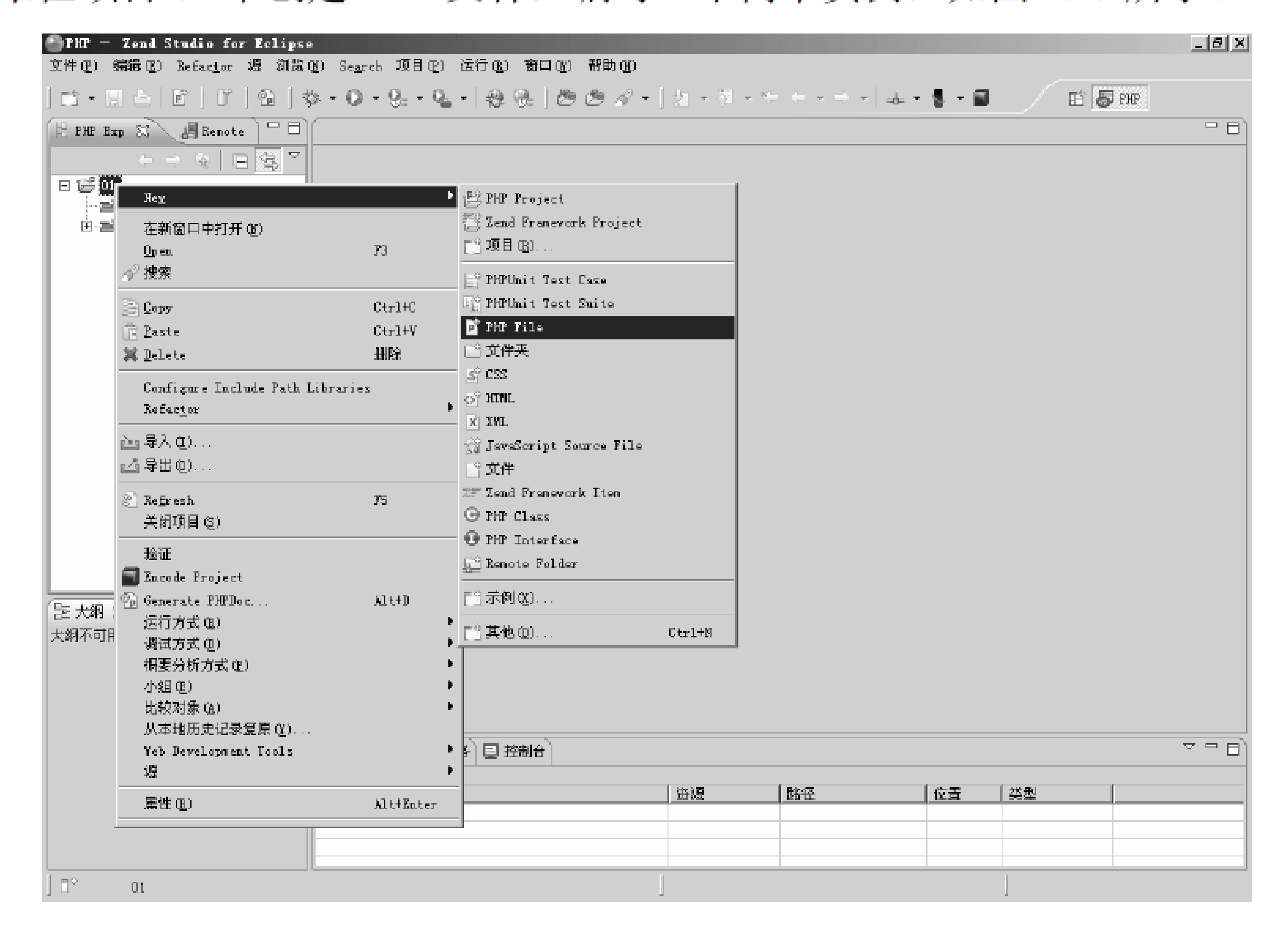


图 2.45　创建 PHP 文件

在这里不仅可以创建 PHP 文件，还可以创建 HTML、CSS、JavaScript 和 XML 文件等，具体可以根据程序的实际情况进行创建。

（7）进入到文件创建窗口，定义文件名称，最后单击“完成”按钮，完成文件的创建，如图 2.46 所示。





图 2.46　创建 index.php 文件

（8）文件创建成功后，即可在编辑区中编辑文件的内容，如图 2.47 所示。



图 2.47　编辑 PHP 代码

（9）编辑完成后，保存该文件，运行程序，运行结果如图 2.48 所示。

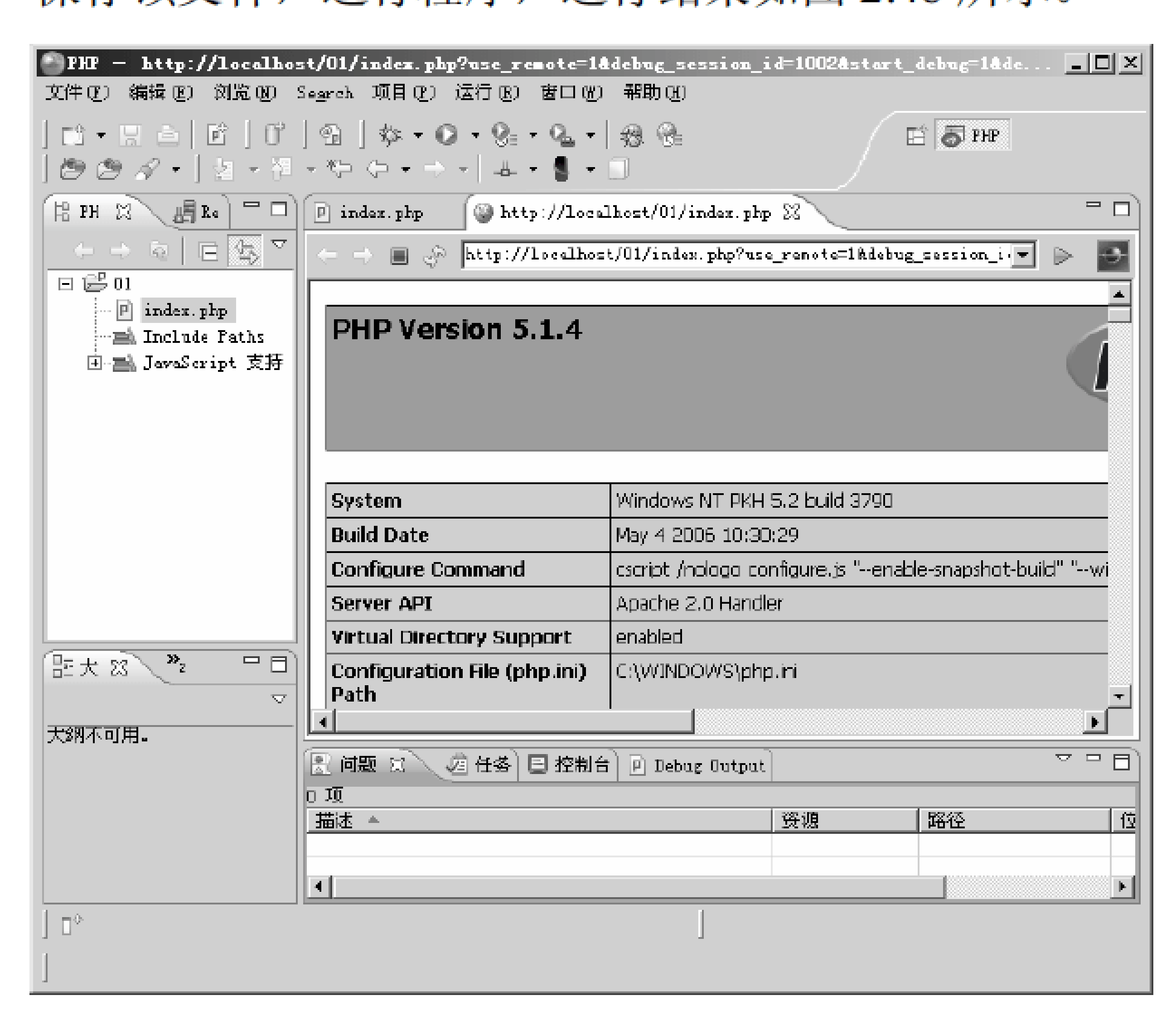


图 2.48　运行程序

2.4.2　Dreamweaver

视频讲解：光盘\TM\lx\2\Dreamweaver 开发工具.exe

Dreamweaver 是 Adobe 公司开发的 Web 站点和应用程序的专业开发工具。它将可视布局工具、应用程序开发功能和代码编辑组合在一起。其功能强大，使得各个层次的设计人员和开发人员都能够美化网站及创建应用程序。从基于 CSS 设计的领先支持到手工编码，Dreamweaver 为专业人员提供了一个集成、高效的环境，这样开发人员可以使用 Dreamweaver 及所选择的服务器来创建功能强大的 Web 应用程序，从而使用户能够连接到数据库、Web 服务和旧式系统。本实例主要讲解如何利用 Dreamweaver 建立站点及开发 PHP 程序。

在 Dreamweaver 中创建站点的操作步骤如下：

（1）选择“站点”→“管理站点”命令，弹出如图 2.49 所示的对话框。



图 2.49　管理 01 站点

（2）单击“管理站点”对话框中的“编辑”按钮，在弹出的“01 的站点定义为”窗口中选择“高级”选项卡，在“分类”列表框中选择“测试服务器”选项，在右侧的“服务器模型”下拉列表框中选择 PHP MySQL 选项，在“访问”下拉列表框中选择“本地/网络”选项，然后设置测试服务器文件夹，也就是指定到站点的根目录下，最后设置 URL 前缀，同样定义到站点的根目录，如图 2.50 所示。

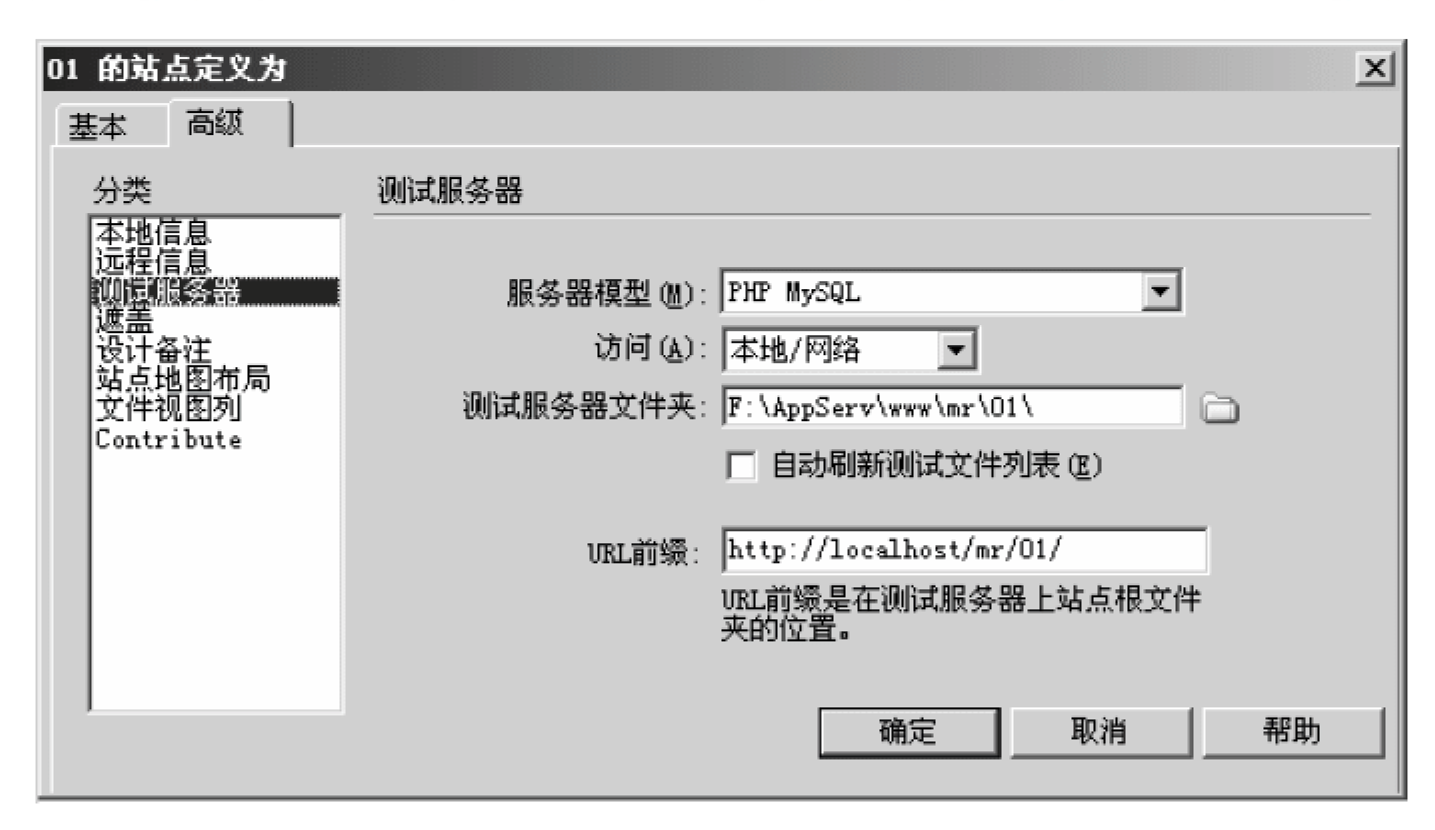
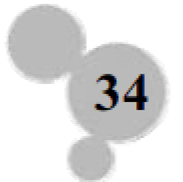


图 2.50　配置测试服务器

（3）单击“确定”按钮，完成站点的设置。在完成测试服务器的配置之后，即可在 Dreamweaver 下直接使用快捷键 F12 来浏览程序。



技巧

在进行站点设置和服务器配置的过程中，必须要将本地的 HTTP 地址与测试服务器中的 URL 前缀统一，都指定到站点的根目录下。例如本实例中，将 HTTP 地址和 URL 前缀都指定到“/mr/01/”文件夹下，其中 mr 是 Apache 服务器根目录下的文件夹，01 则是定义的站点的根目录。

建议 PHP 初学者使用 Dreamweaver 来开发。学习一段时间后，可以再选择另一种开发工具使用。每一种工具都有自己的特点，用户可根据自己的喜好来选择。

2.5　第一个 PHP 实例

视频讲解：光盘\TM\lx\2\第一个 PHP 实例.exe

下面以 Dreamweaver CS3 作为工具开发第一个 PHP 实例。

【例 2.1】 本实例的目的是熟悉 PHP 的书写规则和 Dreamweaver 工具的基本使用。本例的功能很简单，即输出一段欢迎信息。开发步骤如下：**（实例位置：光盘\TM\sl\2\1）**

（1）启动 Dreamweaver。选择“文件”/“新建”命令，或按 Ctrl+N 组合键，弹出“新建文档”对话框。选择“空白页”中的 PHP 页面类型，还可以选择新建页面的布局，这里选择“布局”列表框中的“无”选项。单击“创建”按钮，如图 2.51 所示。



图 2.51　“新建文档”对话框

（2）可以在新创建页面的“代码”视图中编辑 PHP 代码，也可以使用“设计”视图查看 HTML 效果。这里使用“代码”视图，并给该页面设置一个标题，如图 2.52 所示。标题显示的位置在浏览器的左上角，在运行时就能看到效果。

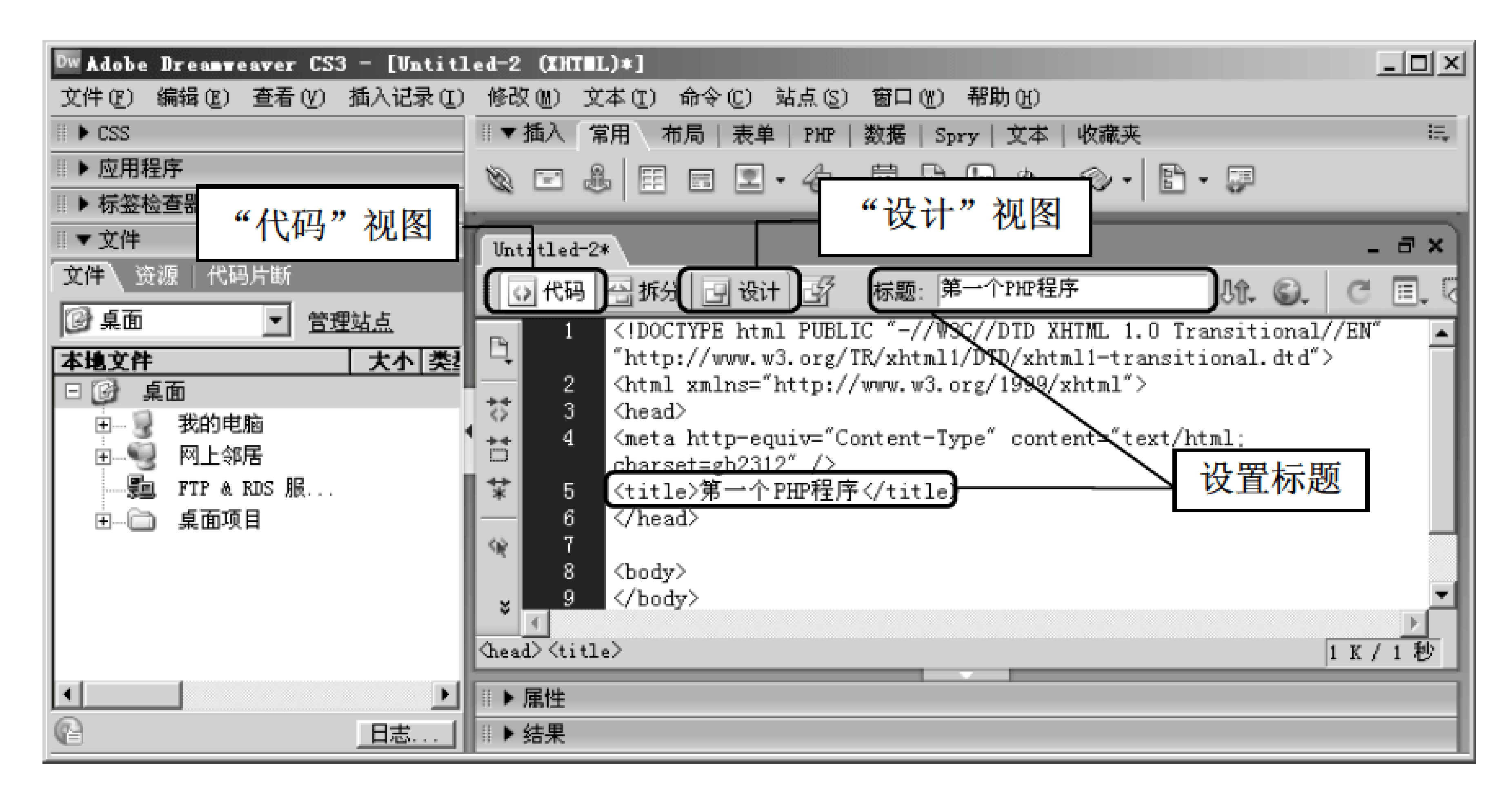


图 2.52　设置标题

（3）编写 PHP 代码。在<body>…</body>标记对中间即可编写 PHP 代码段，实例代码如下：

```
<?php
    echo "欢迎进入 PHP 的世界！！";
?>
```

☑ “<?php”和“?>”是 PHP 的标记对。在这对标记对中的所有代码都被当作 PHP 代码来处理。除了这种表示方法外，PHP 还可以使用 ASP 风格的“<%”和 SGML 风格的“<?…?>”等，在第 3 章中将会详细介绍。

☑ echo 是 PHP 中的输出语句，与 ASP 中的 response.write、JSP 中的 out.print 含义相同，即将紧跟其后的字符串或者变量值显示在页面中。每行代码都以分号“;”结尾。

输入代码的页面如图 2.53 所示。

（4）将 PHP 页保存到服务器指定的目录以便解析。本章中服务器指定的目录为 F:\AppServ\www\。将本页保存到路径 F:\AppServ\www\tm\sl\2\1 下，命名为 index.php。

（5）查看 index.php 页的执行结果。打开 IE 浏览器窗口，在地址栏中输入“http://localhost/tm/sl/2/1/index.php”，按 Enter 键后的页面效果如图 2.54 所示。

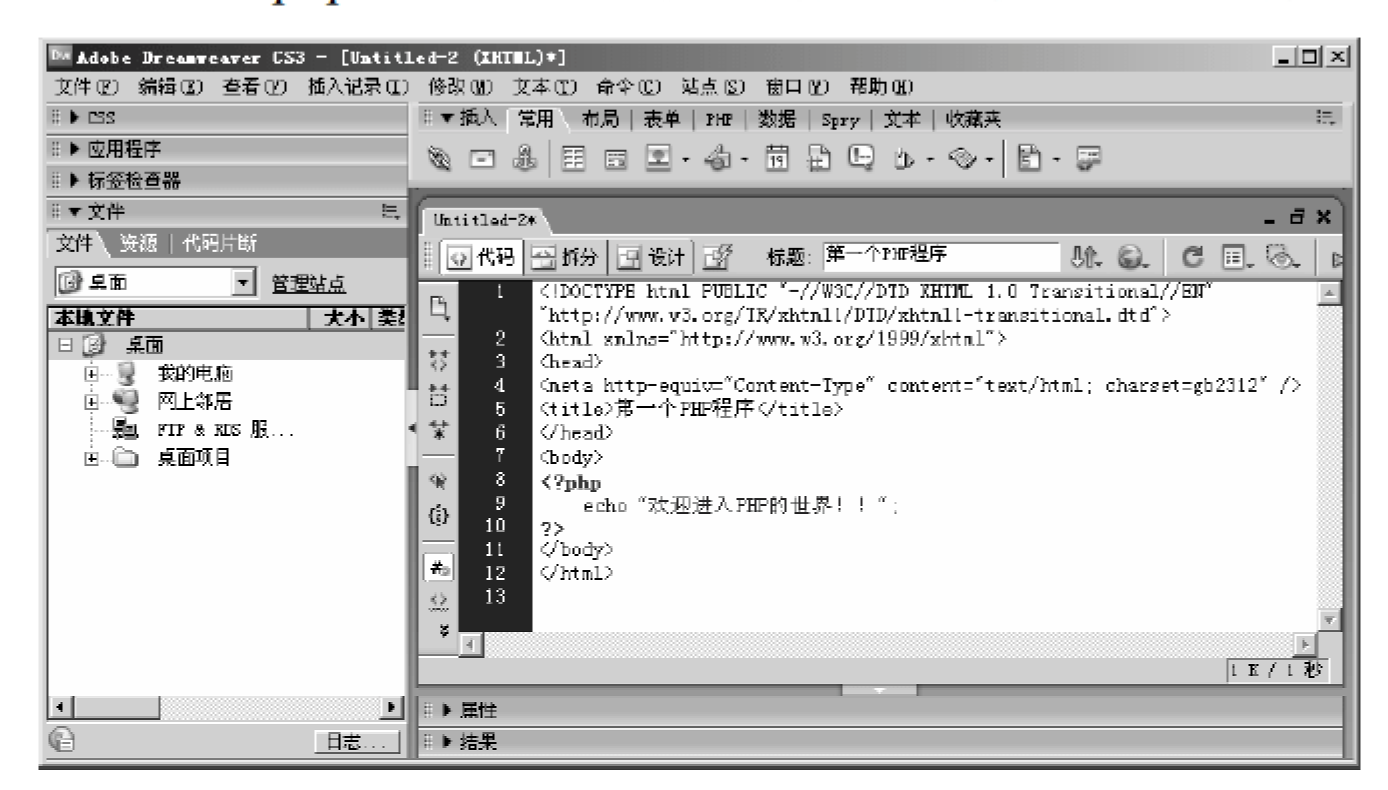
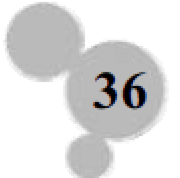


图 2.53　编写程序代码



图 2.54　PHP 页面运行结果



2.6　小　　结

本章主要介绍了在 Windows 和 Linux 下如何搭建 PHP 环境，包括 Apache、PHP 5 和 MySQL 的安装与使用等知识。还介绍了如何让 IIS 支持 PHP 5。除此之外，又介绍了几种方便的组合包和当前比较流行的 PHP 开发工具。希望读者通过本章的学习，能对 PHP 有一个初步的了解，并选择一种适合自己的开发工具。

2.7　练习与实践

1．尝试开发一个页面，使用 echo 语句输出字符串“恭喜您走上 PHP 的编程之路！”。（**答案位置：光盘\TM\sl\2\2**）

2．尝试开发一个页面，使用 echo 语句输出一个 4×3 像素大小的表格。（**答案位置：光盘\TM\sl\2\3**）

第 3 章

PHP 语言基础

（视频讲解：47 分钟）

通过前两章的学习，相信读者对 PHP 的概念和如何搭建 PHP 环境有了一个全面的了解，接下来将学习 PHP 的基础知识。

无论是初出茅庐的“菜鸟”，还是资历深厚的“高手”，没有扎实的基础做后盾是不行的。PHP 的特点是易学、易用，但这并不代表随随便便就可以熟练掌握。随着知识的深入，PHP 会越来越难学，基础的重要性也就越加明显。掌握了基础，就等于有了坚固的地基，才有可能“万丈高楼平地起”。

通过阅读本章，您可以：

- 了解 PHP 的标记风格
- 了解 PHP 的注释种类
- 了解 PHP 的数据类型
- 了解 PHP 的常量变量
- 了解 PHP 运算符
- 了解 PHP 表达式
- 了解 PHP 函数
- 了解 PHP 的编码规范

3.1 PHP 标记风格

视频讲解：光盘\TM\lx\3\PHP 标记和注释.exe

PHP 和其他几种 Web 语言一样，都是使用一对标记对将 PHP 代码部分包含起来，以便和 html 代码相区分。PHP 一共支持 4 种标记风格，下面来一一介绍。

☑ XML 风格

```
<?php
    echo "这是 XML 风格的标记";
?>
```

XML 风格的标记是本书所使用的标记，也是推荐使用的标记，服务器不能禁用。该风格的标记在 XML、XHTML 中都可以使用。

☑ 脚本风格

```
<script language="php">
    echo '这是脚本风格的标记';
</script>
```

☑ 简短风格

```
<? echo '这是简短风格的标记'; ?>
```

☑ ASP 风格

```
<%
  echo '这是 ASP 风格的标记';
%>
```

说明

如果要使用简短风格和 ASP 风格，需要在 php.ini 中对其进行配置，该文件在系统盘 Windows 目录下，如操作系统在 C 盘，那么该文件的位置就是 C:\Windows\php.ini。如果用户用的是 AppServ 安装包，那么通过选择“开始”/“程序”/appServ/Configuration Server/PHP Edit the php.ini configuration File 命令，也可以打开 php.ini 文件，然后将 short_open_tag 和 asp_tags 都设置为 ON，重启 Apache 服务器即可。

注意

这里推荐使用 XML 风格的标记，原因可以参考本章 3.9 节的 PHP 编码规范。



3.2　PHP 注释的应用

视频讲解：光盘\TM\lx\3\PHP 注释的应用.exe

注释即代码的解释和说明，一般放到代码的上方或代码的尾部（放尾部时，代码和注释之间以<tab>键进行分隔，以方便程序阅读），用来说明代码或函数的编写人、用途、时间等。注释不会影响到程序的执行，因为在执行时，注释部分会被解释器忽略不计。

PHP 支持 3 种风格的程序注释。

☑　C++风格的单行注释（//）

```
<?php
echo '使用 C++风格'; //这就是 C++风格
?>
```

☑　C 风格的多行注释（/*…*/）

```
<?php
	/*C
	风格的
	多行注释
	*/
	echo '只会看到这句话。';
?>
```

注意

多行注释是不允许进行嵌套操作的。

☑　Shell 风格的注释（#）

```
<?php
echo '这是 Shell 脚本风格的注释';			#这里的内容是看不到的
?>
```

注意

在单行注释中的内容不要出现“?>”标志，因为解释器会认为 PHP 脚本结束，而去执行“?>”后面的代码。例如：

```
<?php
echo '这样会出错的！！！！！'		//不会看到?>会看到
?>
```

结果为：这样会出错的！！！！！会看到。 ?>

3.3　PHP 的数据类型



视频讲解：光盘\TM\lx\3\PHP 数据类型.exe

PHP 一共支持 8 种原始类型，包括 4 种标量类型，即 boolean（布尔型）、integer（整型）、float/double（浮点型）和 string（字符串型）；两种复合类型，即 array（数组）和 object（对象）；两种特殊类型，即 resource（资源）与 NULL。

说明

PHP 中变量的类型通常不是由程序员设定的，确切地说，是 PHP 根据该变量使用的上下文在运行时决定的。

3.3.1　标量数据类型

标量数据类型是数据结构中最基本的单元，只能存储一个数据。PHP 中标量数据类型包括 4 种，如表 3.1 所示。

表 3.1　标量数据类型

类　型	说　明
boolean（布尔型）	这是最简单的类型。只有两个值，真（true）和假（false）
string（字符串型）	字符串就是连续的字符序列，可以是计算机所能表示的一切字符的集合
integer（整型）	整型数据类型只能包含整数。这些数据类型可以是正数或负数
float（浮点型）	浮点数据类型用于存储数字，和整型不同的是它有小数位

1．布尔型（boolean）

布尔型是 PHP 中较为常用的数据类型之一，它保存一个 true 值或者 false 值，其中 true 和 false 是 PHP 的内部关键字。设定一个布尔型的变量，只需将 true 或者 false 赋值给变量即可。

【例 3.1】 通常布尔型变量都是应用在条件或循环语句的表达式中。下面在 if 条件语句中判断变量$boo 中的值是否为 true，如果为 true，则输出“变量$boo 为真!”，否则输出“变量$boo 为假!!”，实例代码如下：**（实例位置：光盘\TM\sl\3\1）**

```
<?php
	$boo = true;                        //声明一个 boolean 类型变量，赋初值为 true
	if($boo == true)                    //判断变量$boo 是否为真
		echo '变量$boo 为真!';          //为真，输出“变量$boo 为真!”的字样
	else
```



```
        echo '变量$boo 为假!!';            //如果为假，则输出“变量$boo 为假!!”的字样
?>
```

结果为：变量$boo 为真!

注意

在 PHP 中不是只有 false 值才为假的，在一些特殊情况下 boolean 值也被认为是 false。这些特殊情况为：0、0.0、"0"、空白字符串（""）、只声明没有赋值的数组等。

说明

美元符号$是变量的标识符，所有变量都是以$符开头的，无论是声明变量还是调用变量，都应使用$符。

2．字符串型（string）

字符串是连续的字符序列，由数字、字母和符号组成。字符串中的每个字符只占用一个字节。在 PHP 中，有 3 种定义字符串的方式，分别是单引号（'）、双引号（"）和界定符（<<<）。

单引号和双引号是经常被使用的定义方式，定义格式如下：

```
<?php
$a ='字符串';
?>
```

或

```
<?php
$a ="字符串";
?>
```

两者的不同之处在于，双引号中所包含的变量会自动被替换成实际数值，而单引号中包含的变量则按普通字符串输出。

【例 3.2】 下面的实例分别应用单引号和双引号来输出同一个变量，其输出结果完全不同，双引号输出的是变量的值，而单引号输出的是字符串“$i”。实例代码如下：**（实例位置：光盘\TM\sl\3\2）**

```
<?php
    $i = '只会看到一遍';                //声明一个字符串变量
    echo "$i";                          //用双引号输出
    echo "<p>";                         //输出段标记
    echo '$i';                          //用单引号输出
?>
```

运行结果如图 3.1 所示。

两者之间另一处不同点是对转义字符的使用。使用单引号时，只要对单引号“'”进行转义即可，但使用双引号（"）时，还要注意“"”、“$”等字符的使用。这些特殊字符都要通过转义符“\”来显示。常用的转义字符如表 3.2 所示。

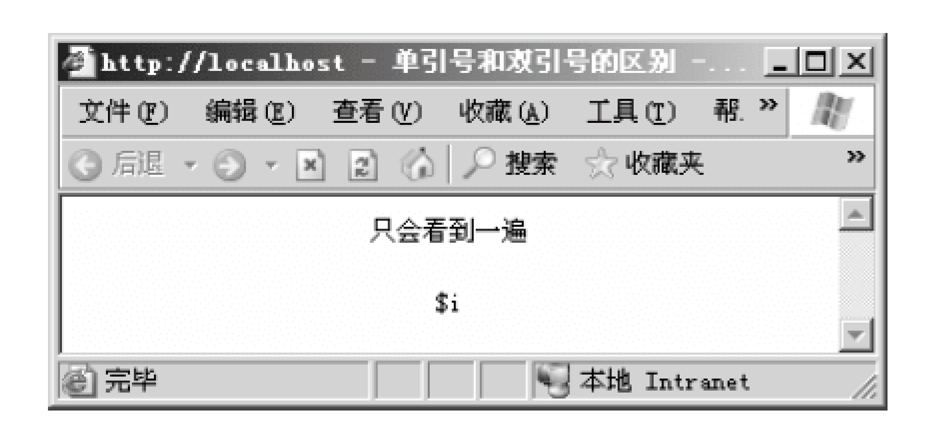


图 3.1　单引号和双引号的区别

表 3.2　转义字符

转 义 字 符	输　　出
\n	换行（LF 或 ASCII 字符 0x0A（10））
\r	回车（CR 或 ASCII 字符 0x0D（13））
\t	水平制表符（HT 或 ASCII 字符 0x09（9））
\\	反斜杠
\$	美元符号
\'	单引号
\"	双引号
\[0-7]{1,3}	此正则表达式序列匹配一个用八进制符号表示的字符，如\467
\x[0-9A-Fa-f]{1,2}	此正则表达式序列匹配一个用十六进制符号表示的字符，如\x9f

\n 和\r 在 Windows 系统中没有什么区别，都可以当作回车符。但在 Linux 系统中则是两种效果，在 Linux 中，\n 表示换到下一行，却不会回到行首；而\r 表示光标回到行首，但仍然在本行。如果读者使用 Linux 操作系统，可以尝试一下。

注意

如果对非转义字符使用了“\”，那么在输出时，“\”也会跟着一起被输出。

说明

在定义简单的字符串时，使用单引号是一个更加合适的处理方式。如果使用双引号，PHP 将花费一些时间来处理字符串的转义和变量的解析。因此，在定义字符串时，如果没有特别的要求，应尽量使用单引号。

界定符（<<<）是从 PHP 4.0 开始支持的。在使用时后接一个标识符，然后是字符串，最后是同样的标识符结束字符串。界定符的格式如下：

```
$string = <<< str
要输出的字符串。
str
```

其中 str 为指定的标识符。

【例 3.3】 下面使用界定符输出变量中的值，可以看到，它和双引号没什么区别，包含的变量也被替换成实际数值，实例代码如下：（**实例位置：光盘\TM\sl\3\3**）



```
<?php
	$i = '显示该行内容';                                            //声明变量$i
	echo <<<std                                                     //界定符开始
	这和双引号没有什么区别，\$i 同样可以被输出出来。<p>              //输出字符串
	\$i 的内容为：$i                                                 //输出变量$i
std;                                                                //界定符结束
?>
```

运行结果如图 3.2 所示。

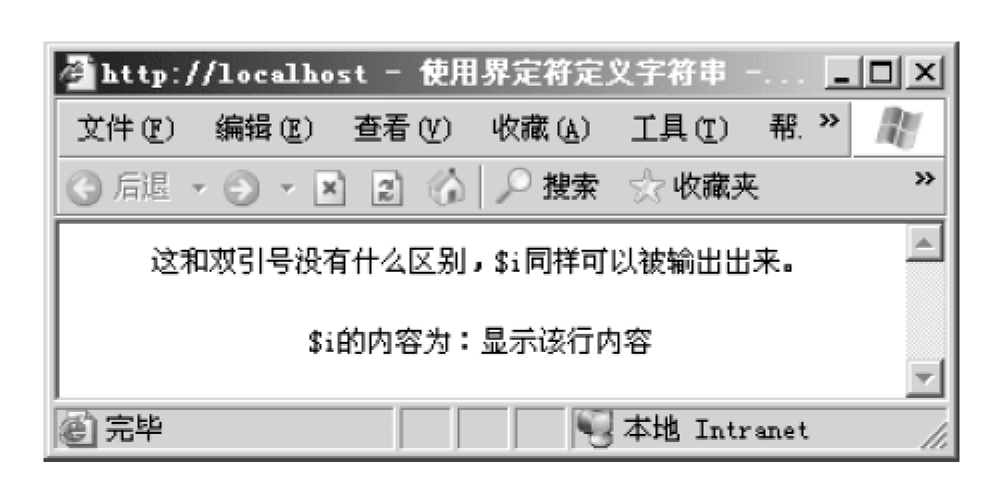


图 3.2　使用界定符定义字符串

注意

结束标识符必须单独另起一行，并且不允许有空格。在标识符前后有其他符号或字符，也会发生错误。例 3.3 中的注释部分在练习时一定不要输入，否则将出现“**Parse error**: parse error, unexpected T_SL in **D:\AppServ\www\tm\sl\3\3\index.php** on line…”的错误提示。

3．整型（integer）

整型数据类型只能包含整数。在 32 位的操作系统中，有效的范围是−2147483648～+2147483647。整型数可以用十进制、八进制和十六进制来表示。如果用八进制，数字前面必须加 0，如果用十六进制，则需要加 0x。

注意

如果在八进制中出现了非法数字（8 和 9），则后面的数字会被忽略掉。

【例 3.4】 本例分别输出八进制、十进制和十六进制的结果，实例代码如下：（**实例位置：光盘\TM\sl\3\4**）

```
<?php
	$str1 = 1234567890;                                  //声明一个十进制的整数
	$str2 = 0x1234567890;                                //声明一个十六进制的整数
	$str3 = 01234567890;                                 //声明一个八进制的整数
	$str4 = 01234567;                                    //声明另一个八进制的整数
	echo '数字 1234567890 不同进制的输出结果：<p>';
	echo '十进制的结果是：'.$str1.'<br>';                 //输出十进制整数
	echo '十六进制的结果是：'.$str2.'<br>';               //输出十六进制整数
	echo '八进制的结果是：';
```

```
if($str3 == $str4){                              //判断$str3 和$str4 的关系
        echo '$str3 = $str4 = '.$str3;           //如果相等，输出变量值
    }else{
        echo '$str3 != str4';                    //如果不相等，输出“$str3 != $str4”
    }
?>
```

运行结果如图 3.3 所示。

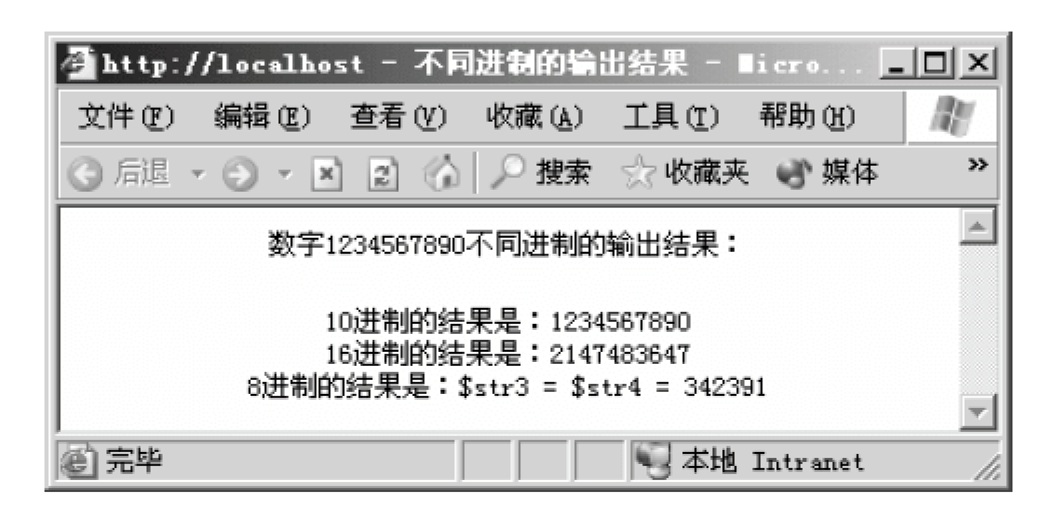


图 3.3　不同进制的输出结果

注意

如果给定的数值超出了 int 型所能表示的最大范围，将会被当作 float 型处理，这种情况称为整数溢出。同样，如果表达式的最后运算结果超出了 int 型的范围，也会返回 float 型。

4．浮点型（float）

浮点数据类型可以用来存储数字，也可以保存小数。它提供的精度比整数大得多。在 32 位的操作系统中，有效的范围是 1.7E−308～1.7E+308。在 PHP 4.0 以前的版本中，浮点型的标识为 double，也叫做双精度浮点数，两者没有区别。

浮点型数据默认有两种书写格式，一种是标准格式：

```
3.1415
-35.8
```

还有一种是科学记数法格式：

```
3.58E1
849.72E-3
```

【例 3.5】 本例中输出圆周率的近似值。用 3 种书写方法：圆周率函数、传统书写格式和科学记数法，最后显示在页面上的效果都一样。实例代码如下：**（实例位置：光盘\TM\sl\3\5）**

```
<?php
echo '圆周率的 3 种书写方法：<p>';
echo '第一种：pi() = '. pi() .'<p>';                                    //调用 pi 函数输出圆周率
echo '第二种：3.14159265359 = '. 3.14159265359 .'<p>';                  //传统书写格式的浮点数
echo '第三种：  314159265359E-11 = '. 314159265359E-11 .'<p>';          //科学记数法格式的浮点数
?>
```

运行结果如图 3.4 所示。

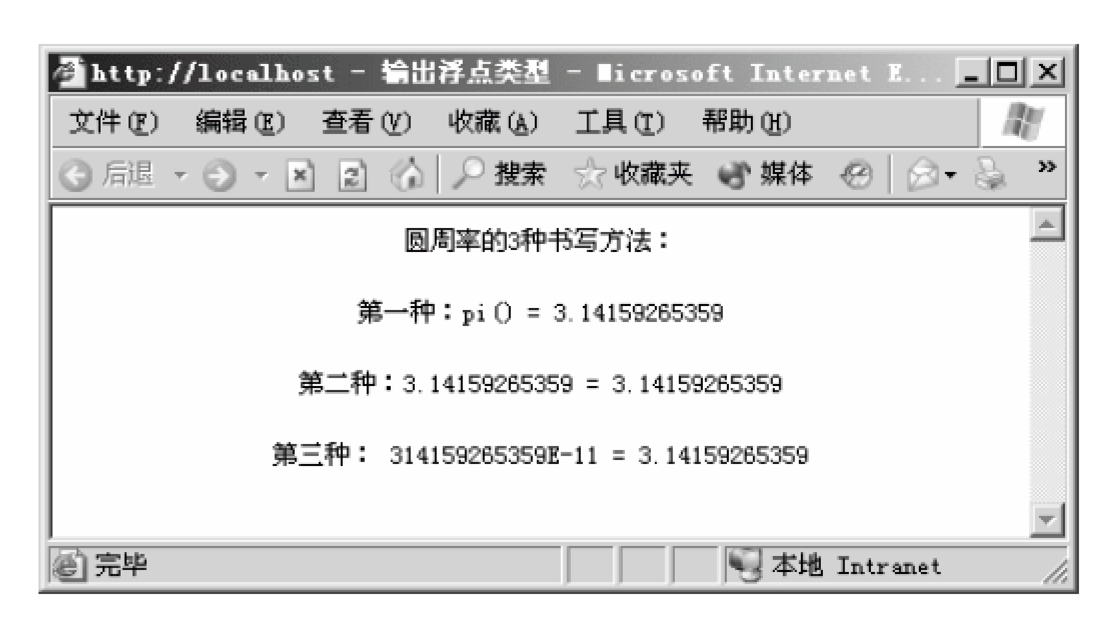


图 3.4　输出浮点类型

注意

浮点型的数值只是一个近似值，所以要尽量避免浮点型数值之间比较大小，因为最后的结果往往是不准确的。

3.3.2　复合数据类型

复合数据类型包括两种，即数组和对象，如表 3.3 所示。

表 3.3　复合数据类型

类　　型	说　　明
array（数组）	一组类型相同的变量的集合
object（对象）	对象是类的实例，使用 new 命令来创建

1. 数组（array）

数组是一组数据的集合，它把一系列数据组织起来，形成一个可操作的整体。数组中可以包括很多数据，如标量数据、数组、对象、资源以及 PHP 中支持的其他语法结构等。

数组中的每个数据称为一个元素，元素包括索引（键名）和值两个部分。元素的索引可以由数字或字符串组成，元素的值可以是多种数据类型。定义数组的语法格式如下：

```
$array = ('value1',' value2 '……)
```

或

```
$array[key] = 'value'
```

或

```
$array = array(key1 => value1, key2 => value2……)
```

其中，参数 key 是数组元素的下标，value 是数组下标所对应的元素。以下几种都是正确的格式：

```
$arr1 = array('This','is','a','example');
$arr2 = array(0 => 'php', 1=>'is', 'the' => 'the', 'str' => 'best ');
$arr3[0] = 'tmpname';
```

声明数组后，数组中的元素个数还可以自由更改。只要给数组赋值，数组就会自动增加长度。在第 7 章 PHP 数组中，会详细介绍数组的使用、取值以及数组的相关函数。

2．对象（object）

编程语言所应用到的方法有两种：面向过程和面向对象。在 PHP 中，用户可以自由使用这两种方法。在第 13 章中将对面向对象的技术进行详细的讲解。

3.3.3　特殊数据类型

特殊数据类型包括资源和空值两种，如表 3.4 所示。

表 3.4　特殊数据类型

类　　型	说　　明
resource（资源）	资源是一种特殊变量，又叫做句柄，保存到外部资源的一个引用。资源是通过专门的函数来建立和使用的
null（空值）	特殊的值，表示变量没有值，唯一的值就是 null

1．资源（resource）

资源类型是 PHP 4 引进的。关于资源的类型，可以参考 PHP 手册后面的附录，里面有详细的介绍和说明。

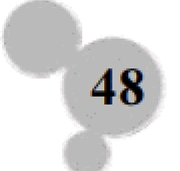
在使用资源时，系统会自动启用垃圾回收机制，释放不再使用的资源，避免内存消耗殆尽。因此，资源很少需要手工释放。

2．空值（null）

空值，顾名思义，表示没有为该变量设置任何值，另外，空值（null）不区分大小写，null 和 NULL 效果是一样的。被赋予空值的情况有以下 3 种：还没有赋任何值、被赋值 null、被 unset()函数处理过的变量。

【例 3.6】 下面来看一个具体实例。字符串 string1 被赋值为 null，string2 根本没有声明和赋值，所以也输出 null，最后的 string3 虽然被赋予了初值，但被 unset()函数处理后，也变为 null 型。unset()函数的作用就是从内存中删除变量。实例代码如下：**（实例位置：光盘\TM\sl\3\6）**

```
<?php
echo "变量(\$string1)直接赋值为 null：";
$string1 = null;                                   //变量$string1 被赋空值
$string3 = "str";                                  //变量$string3 被赋值 str
if(is_null($string1))                              //判断$string1 是否为空
      echo "string1 = null";
```



```
echo "<p>变量(\$string2)未被赋值：";
if(is_null($string2))                                    //判断$string2 是否为空
      echo "string2 = null";
echo "<p>被 unset()函数处理过的变量(\$string3)：";
unset($string3);                                         //释放$string3
if(is_null($string3))                                    //判断$string3 是否为空
      echo "string3 = null";
?>
```

运行结果如图 3.5 所示。

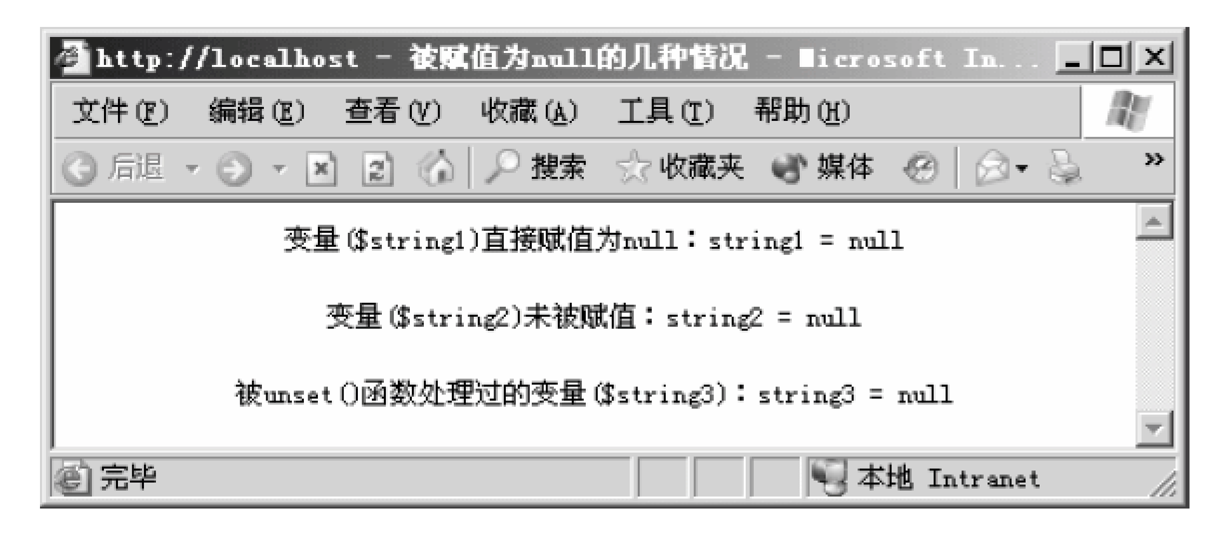


图 3.5　被赋值为 null 的几种情况



is_null()函数是判断变量是否为 null，该函数返回一个 boolean 型，如果变量为 null，则返回 true，否则返回 false。unset()函数用来销毁指定的变量。



从 PHP 4 开始，unset()函数就不再有返回值，所以不要试图获取或输出 unset()。

3.3.4　转换数据类型

虽然 PHP 是弱类型语言，但有时仍然需要用到类型转换。PHP 中的类型转换和 C 语言一样，非常简单，只需在变量前加上用括号括起来的类型名称即可。允许转换的类型如表 3.5 所示。

表 3.5　类型强制转换

转换操作符	转 换 类 型	举　　例
(boolean)	转换成布尔型	(boolean)$num、(boolean)$str
(string)	转换成字符型	(string)$boo、(string)$flo
(integer)	转换成整型	(integer)$boo、(integer)$str
(float)	转换成浮点型	(float)$str、(float)$str
(array)	转换成数组	(array)$str
(object)	转换成对象	(object)$str

注意

在进行类型转换的过程中应该注意以下内容：转换成 boolean 型时，null、0 和未赋值的变量或数组会被转换为 false，其他的为真；转换成整型时，布尔型的 false 转换为 0，true 转换为 1，浮点型的小数部分被舍去，字符型如果以数字开头就截取到非数字位，否则输出 0。

类型转换还可以通过 settype()函数来完成，该函数可以将指定的变量转换成指定的数据类型。

```
bool settype ( mixed var, string type )
```

参数 var 为指定的变量，参数 type 为指定的类型，参数 type 有 7 个可选值，即 boolean、float、integer、array、null、object 和 string。如果转换成功则返回 true，否则返回 false。

当字符串转换为整型或浮点型时，如果字符串是以数字开头的，就会先把数字部分转换为整型，再舍去后面的字符串；如果数字中含有小数点，则会取到小数点前一位。

【例 3.7】 本实例将使用上面的两种方法将指定的字符串进行类型转换，比较两种方法之间的不同。实例代码如下：**（实例位置：光盘\TM\sl\3\7）**

```
<?php
$num = '3.1415926r*r';                                  //声明一个字符串变量
echo '使用(integer)操作符转换变量$num 类型：';
echo (integer)$num;                                     //使用 integer 转换类型
echo '<p>';
echo '输出变量$num 的值：'.$num;                          //输出原始变量$num
echo '<p>';
echo '使用 settype 函数转换变量$num 类型：';
echo settype($num,'integer');                           //使用 settype 函数转换类型
echo '<p>';
echo '输出变量$num 的值：'.$num;                          //输出原始变量$num
?>
```

运行结果如图 3.6 所示。

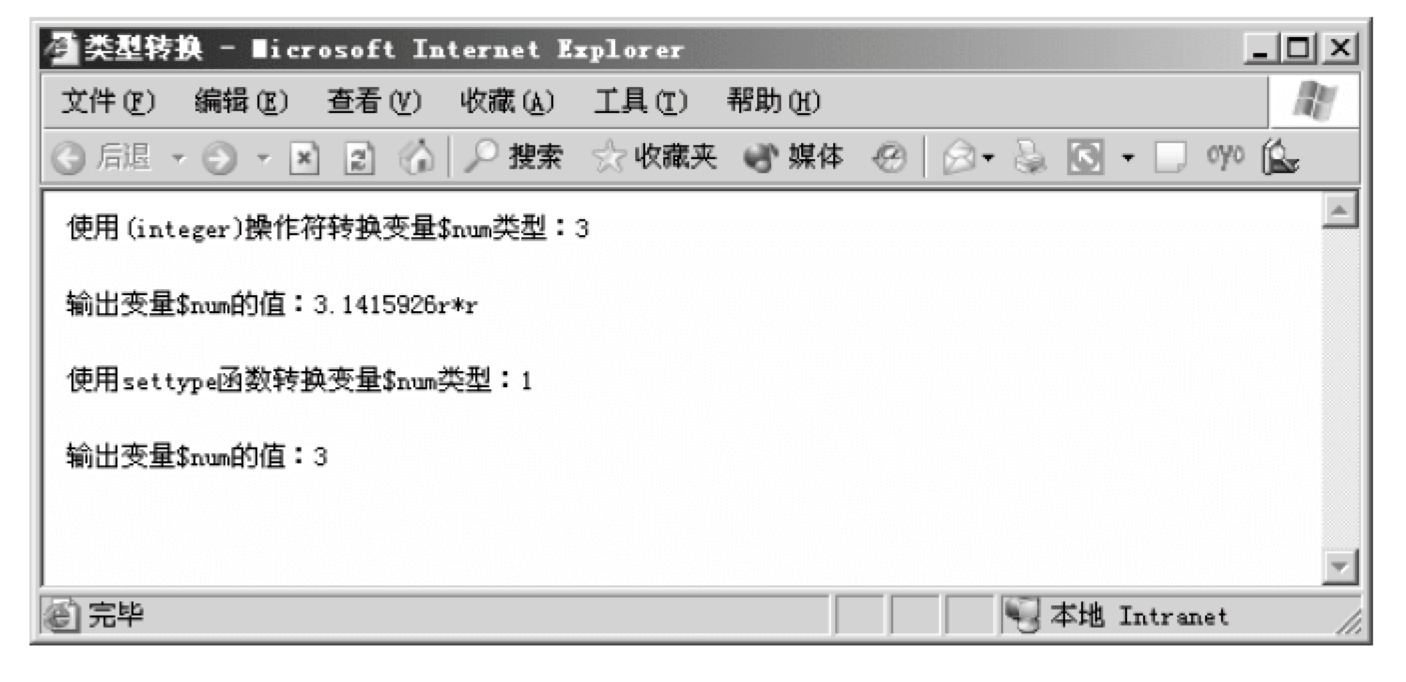
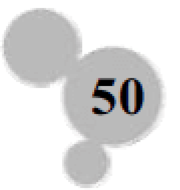


图 3.6 类型转换

可以看到，使用 integer 操作符能直接输出转换后的变量类型。并且原变量不发生任何变化。而使用 settype()函数返回的是 1，也就是 true，而原变量被改变了。在实际应用中，可根据情况自行选择转换方式。



3.3.5　检测数据类型

PHP 还内置了检测数据类型的系列函数，可以对不同类型的数据进行检测，判断其是否属于某个类型，如果符合则返回 true，否则返回 false。检测数据类型的函数如表 3.6 所示。

表 3.6　检测数据类型

函　　数	检 测 类 型	举　　例
is_bool	检查变量是否是布尔类型	is_bool(true)、is_book(false)
is_string	检查变量是否是字符串类型	is_string('string')、is_string(1234)
is_float/is_double	检查变量是否为浮点类型	is_float(3.1415)、is_float('3.1415'))
is_integer/is_int	检查变量是否为整数	is_integer(34)、is_integer('34')
is_null	检查变量是否为 null	is_null(null)
is_array	检查变量是否为数组类型	is_array($arr)
is_object	检查变量是否是一个对象类型	is_object($obj)
is_numeric	检查变量是否为数字或由数字组成的字符串	is_numeric('5')、is_numeric('bccd110')

【例 3.8】 由于检测数据类型的函数的功能和用法都是相同的，下面使用 is_numeric()函数来检测变量中的数据是否是数字，从而了解并掌握 is 系列函数的用法。实例代码如下：**（实例位置：光盘\TM\sl\3\8）**

```
<?php
	$boo = "043112345678";						//声明一个全由数字组成的字符串变量
	if(is_numeric($boo))						//判断该变量是否由数字组成
		echo "Yes,the \$boo a phone number：$boo!";	//如果是，输出该变量
	else
		echo "Sorry,This is an error!";				//否则，输出错误语句
?>
```

结果为：Yes,the $boo a phone number：043112345678

3.4　PHP 常量

视频讲解：光盘\TM\lx\3\常量和变量.exe

本节主要介绍 PHP 常量，包括常量的声明和使用以及预定义常量。

3.4.1　声明和使用常量

常量可以理解为值不变的量。常量值被定义后，在脚本的其他任何地方都不能改变。一个常量由

英文字母、下划线和数字组成，但数字不能作为首字母出现。

在 PHP 中使用 define()函数来定义常量，该函数的语法格式为：

```
define(string constant_name,mixed value,case_sensitive=true)
```

该函数有 3 个参数，详细参数说明如表 3.7 所示。

表 3.7　define 函数的参数说明

参　　数	说　　明
constant_name	必选参数，常量名称，即标识符
value	必选参数，常量的值
case_sensitive	可选参数，指定是否大小写敏感，设定为 true，表示不敏感

获取常量的值有两种方法：一种是使用常量名直接获取值；另一种是使用 constant()函数，constant()函数和直接使用常量名输出的效果是一样的，但函数可以动态地输出不同的常量，在使用上要灵活方便得多。函数的语法格式为：

```
mixed constant(string const_name)
```

参数 const_name 为要获取常量的名称，也可为存储常量名的变量。如果成功则返回常量的值，否则提示错误信息常量没有被定义。

要判断一个常量是否已经定义，可以使用 defined()函数。函数的语法格式为：

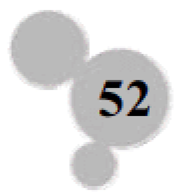
```
bool defined(string constant_name);
```

参数 constant_name 为要获取常量的名称，成功则返回 true，否则返回 false。

【例 3.9】 为了更好地理解如何定义常量，这里给出一个定义常量的实例。在实例中使用上述的 3 个函数：define()函数、constant()函数和 defined()函数。使用 define()函数来定义一个常量，使用 constant()函数来动态获取常量的值，使用 defined()函数来判断常量是否被定义。实例代码如下：（**实例位置：光盘\TM\sl\3\9**）

```
<?php
define ("MESSAGE","能看到一次");
echo MESSAGE."<BR>";                    //输出常量 MESSAGE
echo Message."<BR>";                    //输出“Message”，表示没有该常量
define ("COUNT","能看到多次",true);
echo COUNT."<BR>";                      //输出常量 COUNT
echo Count."<BR>";                      //输出常量 COUNT，因为设定大小写不敏感
$name = "count";
echo constant ($name)."<BR>";           //输出常量 COUNT
echo (defined ("MESSAGE"))."<BR>";      //如果常量被定义，则返回 true，使用 echo 输出显示 1
?>
```

运行结果如图 3.7 所示。



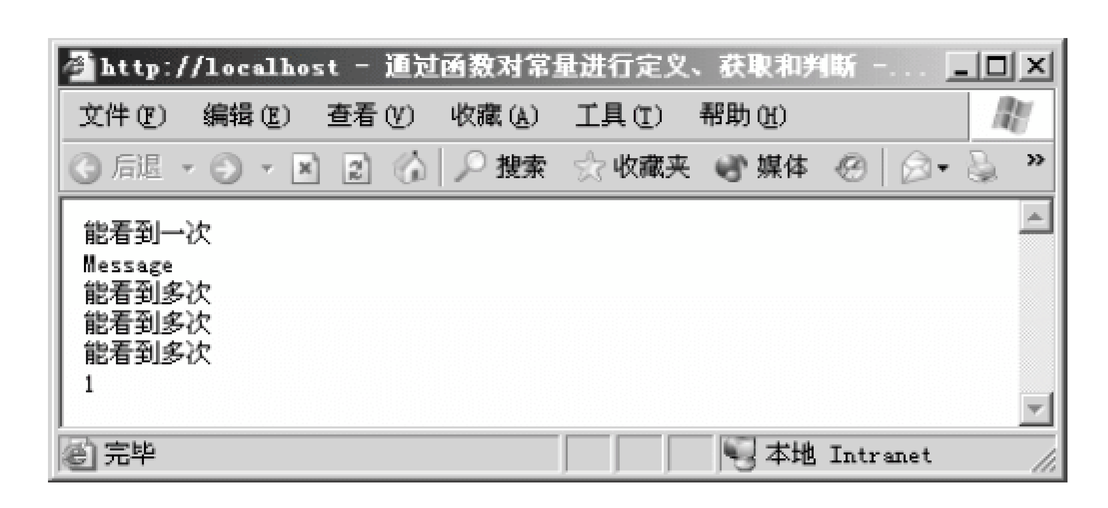


图 3.7　通过函数对常量进行定义、获取和判断

3.4.2　预定义常量

PHP 中可以使用预定义常量获取 PHP 中的信息。常用的预定义常量如表 3.8 所示。

表 3.8　PHP 的预定义常量

常　量　名	功　　能
__FILE__	默认常量，PHP 程序文件名
__LINE__	默认常量，PHP 程序行数
PHP_VERSION	内建常量，PHP 程序的版本，如 3.0.8_dev
PHP_OS	内建常量，执行 PHP 解析器的操作系统名称，如 Windows
TRUE	该常量是一个真值（true）
FALSE	该常量是一个假值（false）
NULL	一个 null 值
E_ERROR	该常量指到最近的错误处
E_WARNING	该常量指到最近的警告处
E_PARSE	该常量指到解析语法有潜在问题处
E_NOTICE	该常量为发生不寻常处的提示但不一定是错误处


注意

__FILE__和__LINE__中的“__”是两条下划线，而不是一条“_”。

说明

表中以 E_开头的预定义常量，是 PHP 的错误调试部分。如需详细了解，请参考 error_ reporting()函数。

【例 3.10】 预定义常量与用户自定义常量在使用上没什么差别。下面使用预定义常量输出 PHP 中的信息。实例代码如下：**（实例位置：光盘\TM\sl\3\10）**

```
<?php
echo "当前文件路径： ".__FILE__;                    //输出__FILE__常量
echo "<br>当前行数：".__LINE__;                      //输出__LINE__常量
echo "<br>当前 PHP 版本信息：".PHP_VERSION;          //输出 PHP 版本信息
```

```
echo "<br> 当前操作系统: ".PHP_OS ;                    //输出系统信息
?>
```

运行结果如图 3.8 所示。

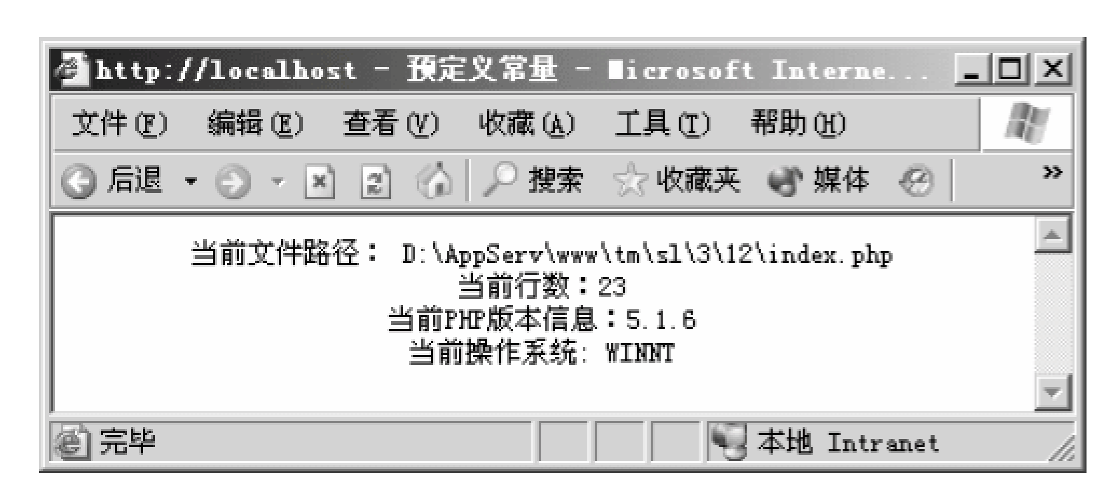


图 3.8　应用 PHP 预定义常量输出信息


说明

根据每个用户操作系统和软件版本的不同，所得的结果也不一定相同。

3.5　PHP 变量

视频讲解：光盘\TM\lx\3\PHP 变量.exe

变量是指在程序执行过程中数值可以变化的量。变量通过一个名字（变量名）来标识。系统为程序中的每一个变量分配一个存储单元，变量名实质上就是计算机内存单元的命名。因此，借助变量名即可访问内存中的数据。

3.5.1　变量声明及使用

和很多语言不同，在 PHP 中使用变量之前不需要声明变量（PHP 4 之前需要声明变量），只需为变量赋值即可。PHP 中的变量名称用$和标识符表示，变量名是区分大小写的。

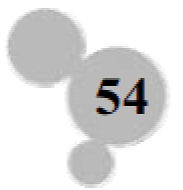
变量赋值，是指给变量一个具体的数据值，对于字符串和数字类型的变量，可以通过“=”来实现。格式为：

```
<?php $name = value; ?>
```

对变量赋值时，要遵循变量命名规则。如下面的变量命名是合法的：

```
<?php
$thisCup="oink";
$_Class="roof ";
?>
```

下面的变量命名则是非法的：



```
<?php
$11112_var=11112;                    //变量名不能以数字字符开头
$@spcn = "spcn";                     //变量名不能以其他字符开头
?>
```

除了直接赋值外，还有两种方式可为变量声明或赋值，一种是变量间的赋值。

【例 3.11】 变量间的赋值是指赋值后两个变量使用各自的内存，互不干扰，实例代码如下：**（实例位置：光盘\TM\sl\3\11）**

```
<?php
$string1 = "spcn";                   //声明变量$string1
$string2 = $string1;                 //使用$string1 初始化$string2
$string1 = "zhuding";                //改变变量$string1 的值
echo $string2;                       //输出变量$string2 的值
?>
```

结果为：spcn

另一种是引用赋值。从 PHP 4 开始，PHP 引入了“引用赋值”的概念。引用的概念是，用不同的名字访问同一个变量内容。当改变其中一个变量的值时，另一个也跟着发生变化。使用&符号来表示引用。

【例 3.12】 在本例中，变量$j 是变量$i 的引用，当给变量$i 赋值后，$j 的值也会跟着发生变化。实例代码如下：**（实例位置：光盘\TM\sl\3\12）**

```
<?php
$i = "spcn";                         //声明变量$i
$j = & $i;                           //使用引用赋值，这时$j 已经赋值为 spcn
$i = "hello ,$i";                    //重新给$j 赋值
echo $j;                             //输出变量$j
echo "<br>";
echo $i;                             //输出变量$i
?>
```

结果为：hello ,spcn　　hello ,spcn



注意

引用和复制的区别在于：复制是将原变量内容复制下来，开辟一个新的内存空间来保存，而引用则是给变量的内容再起一个名字。可以这样理解，一些文学爱好者经常会向报纸、杂志投稿件，但一般不会用真名，而是笔名，这个笔名就可以看作是一个引用。

3.5.2　变量作用域

变量在使用时，要符合变量的定义规则。变量必须在有效范围内使用，如果变量超出有效范围，则变量也就失去其意义了。变量的作用域如表 3.9 所示。

表 3.9　变量作用域

作　用　域	说　　明
局部变量	在函数的内部定义的变量，其作用域是所在函数
全局变量	被定义在所有函数以外的变量，其作用域是整个 PHP 文件，但在用户自定义函数内部是不可用的。如果希望在用户自定义函数内部使用全局变量，则要使用 global 关键字声明
静态变量	能够在函数调用结束后仍保留变量值，当再次回到其作用域时，又可以继续使用原来的值。而一般变量是在函数调用结束后，其存储的数据值将被清除，所占的内存空间被释放。使用静态变量时，先要用关键字 static 来声明变量，把关键字 static 放在要定义的变量之前

在函数内部定义的变量，其作用域为所在函数，如果在函数外赋值，将被认为是完全不同的另一个变量。在退出声明变量的函数时，该变量及相应的值就会被清除。

【例 3.13】本实例用于比较在函数内赋值的变量（局部变量）和在函数外赋值的变量（全局变量），实例代码如下：**（实例位置：光盘\TM\sl\3\13）**

```
<?php
$example="在……函数外";                          //声明全局变量
function example(){
$example="……在函数内……";                        //声明局部变量
echo "在函数内输出的内容是：$example.<br>";        //输出局部变量
}
example();                                       //调用函数，输出变量值
echo "在函数外输出的内容是：$example.<br>";        //输出全局变量
?>
```

运行结果如图 3.9 所示。

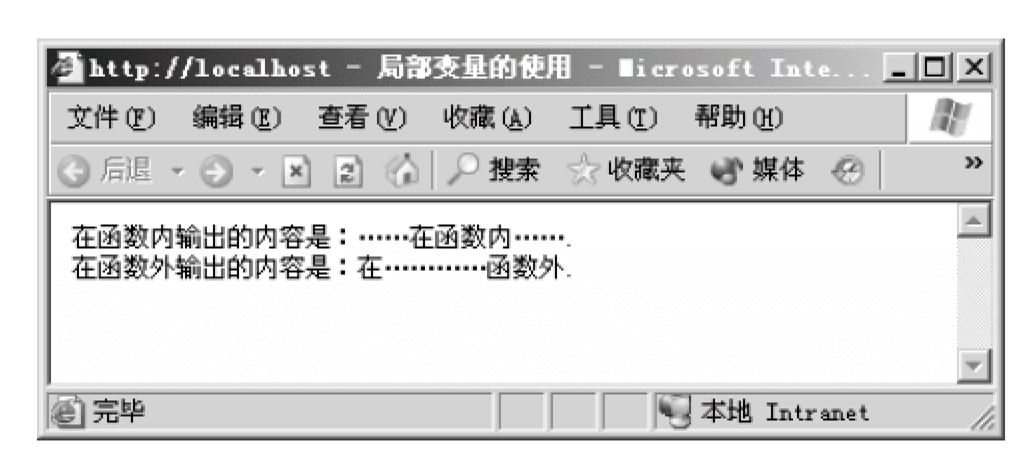


图 3.9　局部变量的使用

静态变量在很多地方都能用到。例如，在博客中使用静态变量记录浏览者的人数，每一次用户访问和离开时，都能够保留目前浏览者的人数。在聊天室中也可以用静态变量来记录用户的聊天内容。

【例 3.14】 下面使用静态变量和普通变量同时输出一个数据，查看一下两者的功能有什么不同。实例代码如下：**（实例位置：光盘\TM\sl\3\14）**

```
<?php
function zdy (){
    static $message = 0 ;             //初始化静态变量
    $message+=1;                      //静态变量加 1
    echo $message." " ;   }           //输出静态变量
function zdy1(){
    $message = 0 ;                    //声明函数内部变量（局部变量）
```



```
    $message += 1 ;                          //局部变量加 1
    echo $message." " ;   }                  //输出局部变量
for ( $i=0 ; $i<10 ; $i++ )      zdy() ;     //输出 1～10
echo "<br>";
for ($i=0 ; $i<10 ; $i++ )      zdy1() ;     //输出 10 个 1
echo "<br>";
?>
```

运行结果如图 3.10 所示。

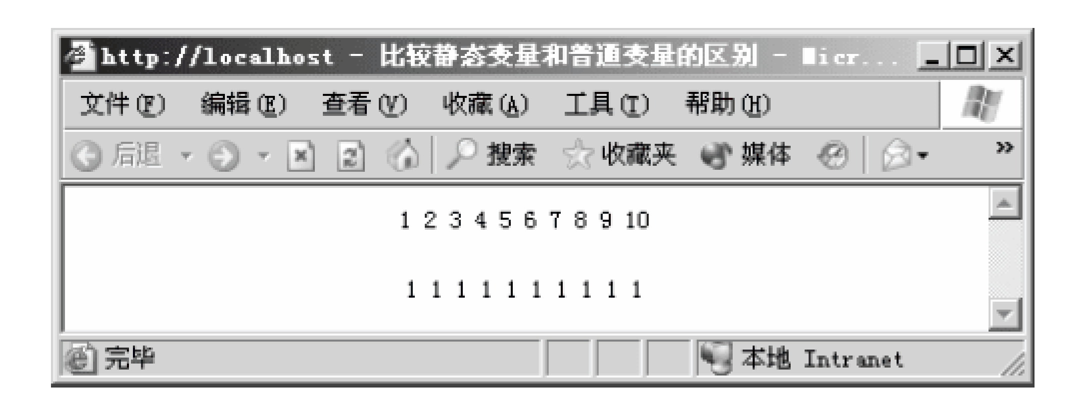


图 3.10　比较静态变量和普通变量的区别

自定义函数 zdy()是输出从 1～10 共 10 个数字，而 zdy1()函数则输出的是 10 个 1。因为自定义函数 zdy()含有静态变量，而函数 zdy1()是一个普通变量。初始化都为 0，再分别使用 for 循环调用两个函数，结果是静态变量的函数 zdy()在被调用后保留了$message 中的值，而静态变量的初始化只是在第一次遇到时被执行，以后就不再对其进行初始化操作了，将会略过第 3 行代码不执行；而普通变量的函数 zdy1()在被调用后，其变量$message 失去了原来的值，重新被初始化为 0。

全局变量可以在程序中的任何地方访问，但是在用户自定义函数内部是不可用的。想在用户自定义函数内部使用全局变量，要使用 global 关键字声明。

【例 3.15】 下面在自定义函数中应用全局变量与不应用全局变量进行对比。本例中定义两个全局变量$zy 和$zyy，在用户自定义函数 lxt()中，希望在第 5、7 行调用它们，而程序输出的结果只有$zyy 的值“会看到”，因为在第 6 行用 global 关键字声明了全局变量$zyy。而第 5 行不会有任何输出，其中的$zy 和第 2 行的$zy 没有任何关系。实例代码如下：**（实例位置：光盘\TM\sl\3\15）**

```
<?php
$zy = "不会看到" ;                  //声明全局变量$zy
$zyy = "会看到" ;                   //声明全局变量$zyy
function lxt(){
      echo $zy."<br>";             //$zy 不能被调用，没有输出
     global $zyy;                  //利用关键字 global 在函数内部定义全局变量
     echo $zyy."<br>" ;            //此处调用$zyy
 }
 lxt() ;
?>
```

结果为：会看到

3.5.3　可变变量

可变变量是一种独特的变量，它允许动态改变一个变量名称。其工作原理是该变量的名称由另外

一个变量的值来确定，实现过程就是在变量的前面再多加一个美元符号“$”。

【例 3.16】 下面使用可变变量动态改变变量的名称。首先定义两个变量$change_name 和$trans，并且输出变量$change_name 的值，然后使用可变变量来改变变量$change_name 的名称，最后输出改变名称后的变量值，实例代码如下：**（实例位置：光盘\TM\sl\3\16）**

```
<?php
$change_name = "trans";          //声明变量$change_name
$trans = "You can see me!";      //声明变量$trans
echo $change_name ;              //输出变量$change_name
echo "<br>" ;
echo $$change_name ;             //通过可变变量输出$trans 的值
?>
```

结果为：trans　　You can see me!

3.5.4　PHP 预定义变量

PHP 还提供了很多非常实用的预定义变量，通过这些预定义变量可以获取到用户会话、用户操作系统的环境和本地操作系统的环境等信息。常用的预定义变量如表 3.10 所示。

表 3.10　预定义变量

变量的名称	说　　明
$_SERVER['SERVER_ADDR']	当前运行脚本所在的服务器的 IP 地址
$_SERVER['SERVER_NAME']	当前运行脚本所在服务器主机的名称。如果该脚本运行在一个虚拟主机上，则该名称是由虚拟主机所设置的值决定
$_SERVER['REQUEST_METHOD']	访问页面时的请求方法。如 GET、HEAD、POST、PUT 等，如果请求的方式是 HEAD，PHP 脚本将在送出头信息后中止（这意味着在产生任何输出后，不再有输出缓冲）
$_SERVER['REMOTE_ADDR']	正在浏览当前页面用户的 IP 地址
$_SERVER['REMOTE_HOST']	正在浏览当前页面用户的主机名。反向域名解析基于该用户的 REMOTE_ADDR
$_SERVER['REMOTE_PORT']	用户连接到服务器时所使用的端口
$_SERVER['SCRIPT_FILENAME']	当前执行脚本的绝对路径名。注意：如果脚本在 CLI 中被执行，作为相对路径，如 file.php 或者../file.php，$_SERVER['SCRIPT_FILENAME']将包含用户指定的相对路径
$_SERVER['SERVER_PORT']	服务器所使用的端口，默认为 80。如果使用 SSL 安全连接，则这个值为用户设置的 HTTP 端口
$_SERVER['SERVER_SIGNATURE']	包含服务器版本和虚拟主机名的字符串
$_SERVER['DOCUMENT_ROOT']	当前运行脚本所在的文档根目录。在服务器配置文件中定义
$_COOKIE	通过 HTTPCookie 传递到脚本的信息。这些 cookie 多数是由执行 PHP 脚本时通过 setcookie()函数设置的
$_SESSION	包含与所有会话变量有关的信息。$_SESSION 变量主要应用于会话控制和页面之间值的传递

续表

变量的名称	说　　明
$_POST	包含通过 POST 方法传递的参数的相关信息。主要用于获取通过 POST 方法提交的数据
$_GET	包含通过 GET 方法传递的参数的相关信息。主要用于获取通过 GET 方法提交的数据
$GLOBALS	由所有已定义全局变量组成的数组。变量名就是该数组的索引。它可以称得上是所有超级变量的超级集合

3.6　PHP 运算符

视频讲解：光盘\TM\lx\3\PHP 运算符及表达式.exe

运算符是用来对变量、常量或数据进行计算的符号，它对一个值或一组值执行一个指定的操作。PHP 的运算符包括算术运算符、字符串运算符、赋值运算符、位运算符、逻辑运算符、比较运算符、递增或递减运算符、错误控制运算符，这里只介绍一些常用的运算符。

3.6.1　算术运算符

算术运算（Arithmetic Operators）符号是处理四则运算的符号。在数字的处理中应用得最多。常用的算术运算符如表 3.11 所示。

表 3.11　常用的算术运算符

名　　称	操　作　符	举　　例
加法运算	+	$a + $b
减法运算	-	$a-$b
乘法运算	*	$a * $b
除法运算	/	$a / $b
取余数运算	%	$a % $b
递增运算	++	$a++、++$a
递减运算	--	$a--、--$a



说明

在算术运算符中使用%求余，如果被除数（$a）是负数，那么取得的结果也是一个负值。

最后两个递增/递减运算符，主要是对单独一个变量来操作的。递增/递减运算符有两种使用方法，一种是先将变量增加或者减少 1，然后再将值赋给原变量，称为前置递增或递减运算符；另一种是将运

算符放在变量后面，即先返回变量的当前值，然后再将变量的当前值增加或者减少 1，称为后置递增或递减运算符。

【例 3.17】本例分别使用上述几种运算符进行运算，实例代码如下：（**实例位置：光盘\TM\sl\3\17**）

```
<?php
    $a = -100;                              //声明变量$a
    $b = 50;                                //声明变量$b
    $c = 30;                                //声明变量$c
    echo "\$a = ".$a.",";                   //输出变量
    echo "\$b = ".$b.",";
    echo "\$c = ".$c."<p>";                 //输出变量 c
    echo "\$a + \$b = ".($a + $b)."<br>";   //计算变量$a 加$b 的值
    echo "\$a - \$b = ".($a - $b)."<br>";;  //计算变量$a 减$b 的值
    echo "\$a * \$b = ".($a * $b)."<br>";   //计算$a 乘$b 的值
    echo "\$a / \$b = ".($a / $b)."<br>";   //计算$a 除以$b 的值
    echo "\$a % \$c = ".($a % $c)."<br>";   //计算$a 和$b 的余数，被除数为-100
    echo "\$a++ = ".$a++." ";               //对变量$a 进行后置递增运算
    echo "运算后\$a 的值为：".$a."<br>";
    echo "\$b-- = ".$b--." ";               //对变量$b 进行后置递减运算
    echo "运算后\$b 的值为：".$b."<br>";
    echo "++\$c = ".++$c." ";               //对变量$c 进行前置递增运算
    echo "运算后\$c 的值为：".$c;
?>
```

运行结果如图 3.11 所示。

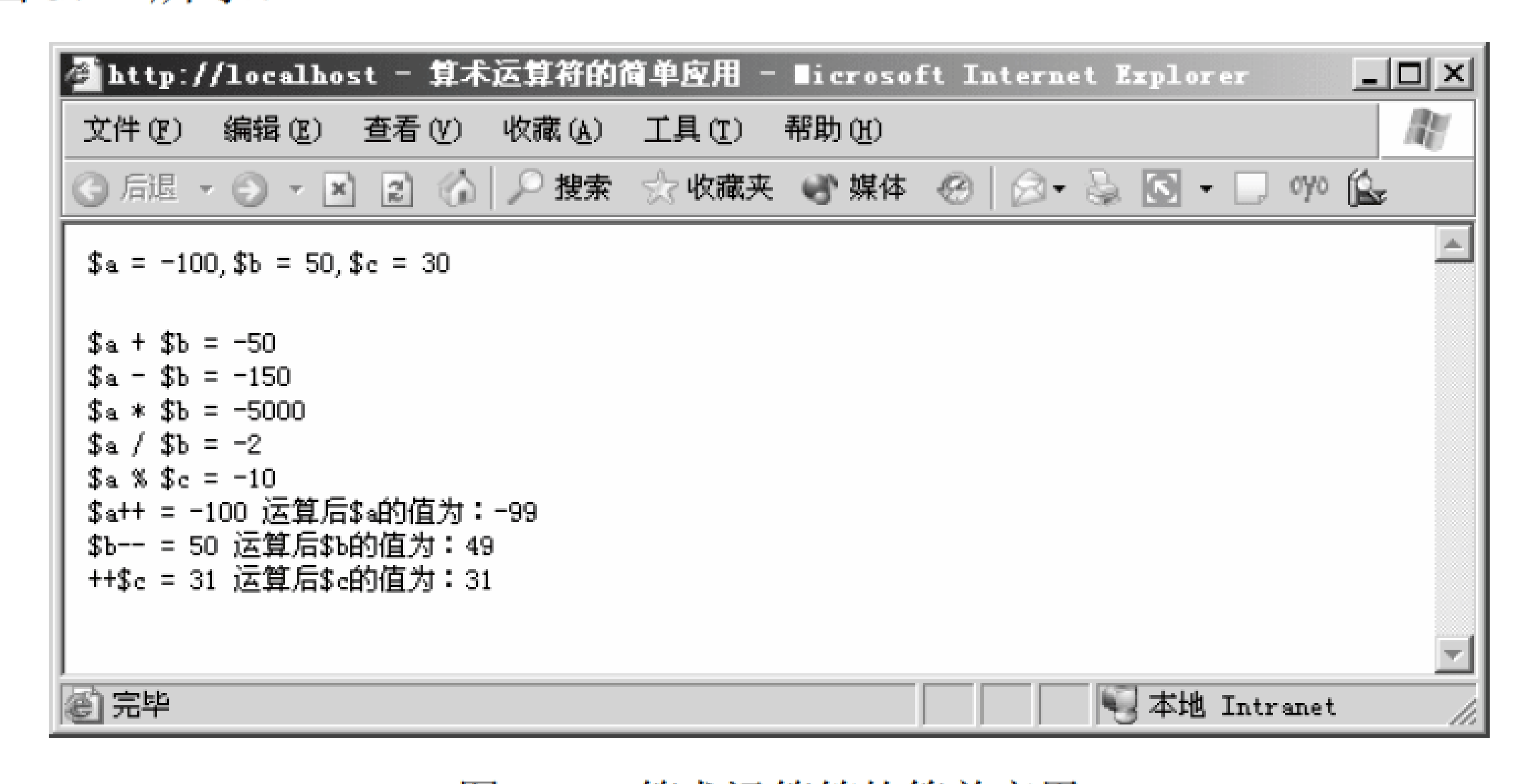
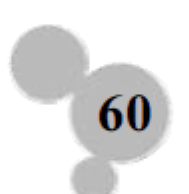


图 3.11　算术运算符的简单应用

3.6.2　字符串运算符

字符串运算符只有一个，即英文的句号“.”。它将两个字符串连接起来，结合成一个新的字符串。使用过 C 或 Java 的读者应注意，这里的“+”号只用作赋值运算符使用，而不能用作字符串运算符。

【例 3.18】 本实例用于对比“.”和“+”两者之间的区别。当使用“.”时，变量$m 和$n 两个字符串组成一个新的字符串 3.1415926r*r1，当使用“+”时，PHP 会认为这是一次运算。如果“+”号的



两边有字符类型，则自动转换为整型；如果是字母，则输出为 0；如果是以数字开头的字符串，则会截取字串头部的数字，再进行运算。实例代码如下：**（实例位置：光盘\TM\sl\3\18）**

```
<?php
	$n = "3.1415926r*r";			//声明一个字符串变量，以数字开头
	$m = 1;					//声明一个整型变量
	$nm = $n.$m;				//使用“.”运算符将两个变量连接
	echo $nm."<br>";
	$mn = $n + $m ;				//使用“+”运算符将两个变量连接
	echo $mn . "<br>" ;
?>
```

结果为：3.1415926r*r1　　4.1415926

3.6.3　赋值运算符

赋值运算符是把基本赋值运算符“=”右边的值赋给左边的变量或者常量。在 PHP 中的赋值运算符如表 3.12 所示。

表 3.12　常用赋值运算符

操　作	符　号	举　例	展 开 形 式	意　义
赋值	=	$a=b	$a=3	将右边的值赋给左边
加	+=	$a+= b	$a=$a+b	将右边的值加到左边
减	-=	$a-= b	$a=$a-b	将右边的值减到左边
乘	*=	$a*= b	$a=$a * b	将左边的值乘以右边
除	/=	$a/= b	$a=$a / b	将左边的值除以右边
连接字符	.=	$a.= b	$a=$a. b	将右边的字符加到左边
取余数	%	$a%= b	$a=$a % b	将左边的值对右边取余数

3.6.4　位运算符

位逻辑运算符是指对二进制位从低位到高位对齐后进行运算。在 PHP 中的位运算符如表 3.13 所示。

表 3.13　位运算符

符　号	作　用	举　例
&	按位与	$m & $n
|	按位或	$m | $n
^	按位异或	$m ^ $n
～	按位取反	$m～$n
<<	向左移位	$m << $n
>>	向右移位	$m >> $n

【例 3.19】 下面使用位运算符对变量中的值进行位运算操作，实例代码如下：（**实例位置：光盘\TM\sl\3\19**）

```
<?php
$m = 8 ;
$n = 12 ;
$mn = $m & $n ;                    //位与
echo $mn ."<br>";
$mn = $m | $n ;                    //位或
echo $mn ."<br>";
$mn = $m ^ $n ;                    //位异或
echo $mn ."<br>";
$mn =  ~$m ;                       //位取反
echo $mn ."<br>";
?>
```

结果为：8　　12　4　 −9

3.6.5　逻辑运算符

逻辑运算符用来组合逻辑运算的结果，是程序设计中一组非常重要的运算符。PHP 的逻辑运算符如表 3.14 所示。

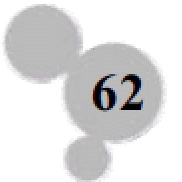
表 3.14　PHP 的逻辑运算符

运　算　符	举　　例	结 果 为 真
&&或 and（逻辑与）	$m and $n	当$m 和$n 都为真时
||或 or（逻辑或）	$m || $n	当$m 为真或者$n 为真时
xor（逻辑异或）	$m xor $n	当$m、$n 一真一假时
!（逻辑非）	!$m	当$m 为假时

在逻辑运算符中，逻辑与和逻辑或这两个运算符有 4 种运算符号（&&、and、||和 or），其中属于同一个逻辑结构的两个运算符号（例如&&和 and）之间却有着不同的优先级。

【例 3.20】 本实例分别使用逻辑或中的运算符号“||”和“or”进行相同的判断，因为同一逻辑结构的两个运算符“||”和“or”的优先级不同，输出的结果也不同。实例代码如下：（**实例位置：光盘\TM\sl\3\20**）

```
<?php
    $i = true;                    //声明一个布尔型变量$i，赋值为真
    $j = true;                    //声明一个布尔型变量$j，赋值也为真
    $z = false;                   //声明一个初值为假的布尔变量$z
    if($i or $j and $z)           //用 or 作判断
        echo "true";              //如果 if 表达式为真，输出 true
    else
        echo "false";             //否则输出 false
    echo "<br>";
```



```
    if($i || $j and $z)           //用||作判断
        echo "true";              //如果表达式为真，输出 true
    else
        echo "false";             //如果表达式为假，输出 false
?>
```

结果为：true false



注意

可以看到，两个 if 语句除了 or 和||不同之外，其他完全一样，但最后的结果却正好相反。在实际应用中要多注意一下这样的细节。

3.6.6　比较运算符

比较运算符就是对变量或表达式的结果进行大小、真假等比较，如果比较结果为真，则返回 true，如果为假，则返回 false。PHP 中的比较运算符如表 3.15 所示。

表 3.15　PHP 的比较运算符

运　算　符	说　　明	举　　例
<	小于	$m<$n
>	大于	$m>$n
<=	小于等于	$m<=$n
>=	大于等于	$m>=$n
= =	相等	$m= =$n
!=	不等	$m!=$n
= = =	恒等	$m= = = $n
!==	非恒等	$m!==$n

其中，不太常见的就是= = =和!= =。$a = = = $b，说明$a 和$b 不只是数值上相等，而且两者的类型也一样。!= =和= = =的意义相近，$a != = $b 就是说$a 和$b 或者数值不等，或者类型不等。

【例 3.21】 本实例使用比较运算符对变量中的值进行比较，设置变量$value =“100”，变量的类型为字符串型，将变量$value 与数字 100 进行比较，会发现比较的结果非常有趣。其中使用的 var_dump 函数是系统函数，作用是输出变量的相关信息。实例代码如下：**（实例位置：光盘\TM\sl\3\21）**

```
<?PHP
$value="100";                     //声明一个字符串变量$value
echo "\$value = \"$value\"";
echo "<p>\$value==100: ";
var_dump($value==100);            //结果为:bool(true)
echo "<p>\$value==ture: ";
var_dump($value==true);           //结果为:bool(true)
echo "<p>\$value!=null: ";
var_dump($value!=null);           //结果为:bool(true)
```

```
echo "<p>\$value==false: ";
var_dump($value==false);            //结果为:bool(false)
echo "<p>\$value === 100: ";
var_dump($value===100);             /结果为:bool(false)
echo "<p>\$value===true: ";
var_dump($value===true);            //结果为:bool(true)
echo "<p>(10/2.0 !== 5): ";
var_dump(10/2.0 !==5);              //结果为:bool(true)
?>
```

运行结果如图 3.12 所示。



图 3.12　比较运算符的应用

3.6.7　错误控制运算符

@错误屏蔽运算符可以对程序中出现错误的表达式进行操作，进而对错误信息进行屏蔽，其使用的方法就是在错误的表达式前加上@即可。@只是对错误信息进行屏蔽，并没有真正解决错误。

经常在程序中使用的某些函数出现一些不必要（不影响程序运行的错误）的错误信息时，使用该运算符进行屏蔽。针对程序中的一些影响程序运行的错误，使用它不是解决问题的根本办法，不推荐使用。

下面了解一下该运算符的使用方法。在进行数学计算时会发生一些错误，例如：

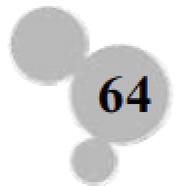
```
<?php
    $err = 5 / 0;
?>
```

这时屏幕上会显示错误消息：Warning: Division by zero in ……\index.php on line 9

如果不想显示这个错误，就可以在表达式前加上@，实现代码如下：

```
<?php
$err = @(5 / 0);
?>
```

这样，输出时就什么错误都不显示。当然，错误依然存在，只是看不到而已。



3.6.8　三元运算符

三元运算符（?:），也称为三目运算符，用于根据一个表达式在另两个表达式中选择一个，而不是用来在两个语句或者程序中选择。三元运算符最好放在括号里使用。

【例 3.22】 下面应用三元运算符实现一个简单的判断功能，如果正确则输出“三元运算”，否则输出“没有该值”，实例代码如下：**（实例位置：光盘\TM\sl\3\22）**

```
<?php
$value=100;                                    //声明一个整型变量
echo ($value==true)?三元运算: 没有该值;           //对整型变量进行判断
?>
```

结果为：三元运算

3.6.9　运算符的优先顺序和结合规则

所谓运算符的优先级，是指在应用中哪一个运算符先计算，哪一个后计算，与数学的四则运算遵循的“先乘除，后加减”是一个道理。

PHP 的运算符在运算中遵循的规则是：优先级高的运算先执行，优先级低的操作后执行，同一优先级的操作按照从左到右的顺序进行。也可以像四则运算那样使用小括号，括号内的运算最先进行。PHP 运算符优先级如表 3.16 所示。

表 3.16　运算符的优先级

优 先 级 别	运　算　符
1	or，and，xor
2	赋值运算符
3	||，&&
4	|，^
5	&，.
6	+，-（递增或递减运算符）
7	/，*，%
8	<<，>>
9	++，--
10	+，-（正、负号运算符），!，～
11	= =，!=，<>
12	<，<=，>，>=
13	?:
14	—>
15	=>

这么多的级别，如果想都记住是不太现实的，也没有必要。如果写的表达式真的很复杂，而且包

含了较多的运算符，不妨多使用括号，例如：

```
<?php
	$a and (($b != $c) or (5 * (50 – $d)))
?>
```

这样就会减少出现逻辑错误的可能。

3.7　PHP 的表达式

视频讲解：光盘\TM\lx\3\PHP 的表达式.exe

表达式是构成 PHP 程序语言的基本元素，也是 PHP 最重要的组成元素。在 PHP 语言中，几乎所写的任何对象都是表达式。最基本的表达式形式是常量和变量。如$m=20，即表示将值 20 赋给变量$m。

表达式是通过具体的代码来实现的，是多个符号集合起来组成的代码，而这些符号只是一些对 PHP 解释程序有具体含义的最小单元。它们可以是变量名、函数名、运算符、字符串、数值和括号等。如以下代码：

```
<?PHP
"fine" ;
$a = "word" ;
?>
```

这就是由两个表达式组成的脚本，即 fine 和$a="word"。此外，还可以进行连续赋值，如：

```
<?php
	$b = $a =5;
?>
```

因为 PHP 赋值操作的顺序是由右到左的，所以变量$b 和$a 都被赋值 5。

在 PHP 的代码中，使用分号“;”来区分表达式，表达式也可以包含在括号内。可以这样理解：一个表达式再加上一个分号，就是一条 PHP 语句。

应用表达式能够做很多事情，如调用一个数组、创建一个类、给变量赋值等。

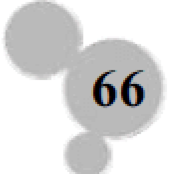

注意

在编写程序时，应该注意表达式后面的分号“;”不要漏写。

3.8　PHP 函数

视频讲解：光盘\TM\lx\3\PHP 函数.exe

在开发过程中，经常要反复重复某种操作或处理，如数据查询、字符操作等，如果每个模块的操



作都要重新输入一次代码，不仅令程序员头痛不已，而且对于代码的后期维护及运行效果也有着较大的影响，使用 PHP 函数即可让这些问题迎刃而解，下面即介绍这些知识。

3.8.1　定义和调用函数

函数，就是将一些重复使用到的功能写在一个独立的代码块中，在需要时单独调用。创建函数的基本语法格式为：

```
function fun_name($str1,$stgr2…$strn){
	fun_body;
}
```

其中，

☑　function：为声明自定义函数时必须使用到的关键字。
☑　fun_name：为自定义函数的名称。
☑　$str1…$strn：为函数的参数。
☑　fun_body：为自定义函数的主体，是功能实现部分。

当函数被定义好后，所要做的就是调用这个函数。调用函数的操作十分简单，只需要引用函数名并赋予正确的参数即可完成函数的调用。

【例 3.23】 在本例中定义了一个函数 example()，计算传入的参数的平方，然后连同表达式和结果全部输出。实例代码如下：（**实例位置：光盘\TM\sl\3\23**）

```
<?php
	/*	声明自定义函数		*/
	function example($num){
		return "$num * $num = ".$num * $num;			//返回计算后的结果
	}
	echo example(10);						//调用函数
?>
```

结果为：10 * 10 = 100

3.8.2　在函数间传递参数

在调用函数时，需要向函数传递参数，被传入的参数称为实参，而函数定义的参数为形参。参数传递的方式有按值传递、按引用传递和默认参数 3 种。

1．按值传递方式

将实参的值复制到对应的形参中，在函数内部的操作针对形参进行，操作的结果不会影响到实参，即函数返回后，实参的值不会改变。

【例 3.24】 本例首先定义一个函数 example()，功能是将传入的参数值做一些运算后再输出。接着在函数外部定义一个变量$m，也就是要传进来的参数。最后调用函数 example($m)，输出函数的返

回值$m 和变量$m 的值。实例代码如下：**（实例位置：光盘\TM\sl\3\24）**

```
<?php
function example( $m ){                     //定义一个函数
      $m = $m * 5 + 10;
echo "在函数内：\$m = ".$m;                  //输出形参的值
}
$m = 1;
example( $m ) ;                             //传递值，将$m 的值传递给形参$m
echo "<p>在函数外  \$m = $m <p>" ;           //实参的值没有发生变化，输出 m=1
?>
```

运行结果如图 3.13 所示。

2．按引用传递方式

按引用传递就是将实参的内存地址传递到形参中。这时，在函数内部的所有操作都会影响到实参的值，返回后，实参的值会发生变化。引用传递方式就是传值时在原基础上加&号即可。

【例 3.25】 仍然使用例 3.24 中的代码，唯一不同的地方就是多了一个&号。实例代码如下：**（实例位置：光盘\TM\sl\3\25）**

```
<?php
function example( &$m ){                    //定义一个函数，同时传递参数$m 的变量
      $m = $m * 5 + 10;
echo "在函数内：\$m = ".$m;                  //输出形参的值
}
$m = 1;
example( $m ) ;                             //传递值：将$m 的值传递给形参$m
echo "<p>在函数外：\$m = $m <p>" ;          //实参的值发生变化，输出 m=15
?>
```

运行结果如图 3.14 所示。

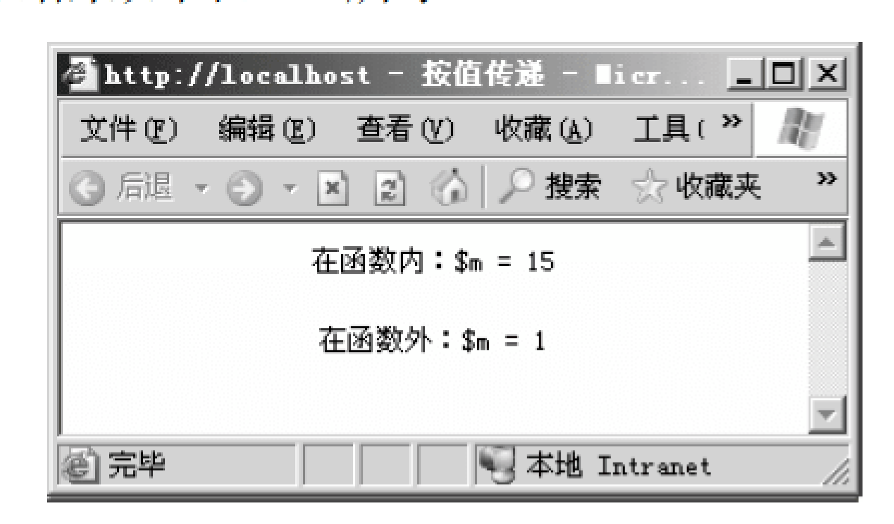


图 3.13　按值传递

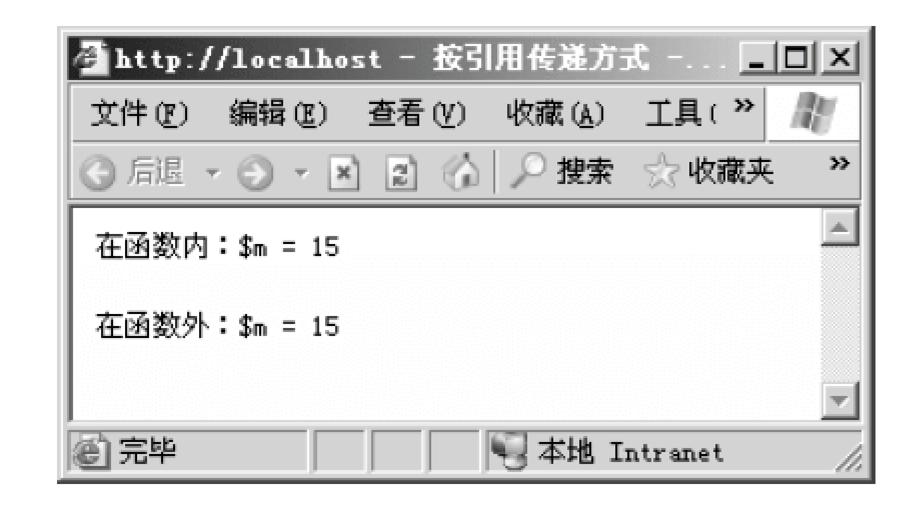
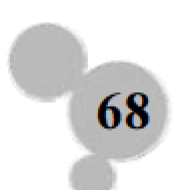


图 3.14　按引用传递方式

3．默认参数（可选参数）

还有一种设置参数的方式，即可选参数。可以指定某个参数为可选参数，将可选参数放在参数列表末尾，并且指定其默认值为空。

【例 3.26】 本实例使用可选参数实现一个简单的价格计算功能，设置自定义函数 values 的参数$tax 为可选参数，其默认值为空。第一次调用该函数，并且给参数$tax 赋值 0.25，输出价格；第二次



调用该函数，不给参数$tax 赋值，输出价格。实例代码如下：**（实例位置：光盘\TM\sl\3\26）**

```
<?php
    function values($price,$tax=""){        //定义一个函数，其中的一个参数初始值为空
        $price=$price+($price*$tax);        //声明一个变量$price，等于两个参数的运算结果
        echo "价格:$price<br>";              //输出价格
    }
    values(100,0.25);                        //为可选参数赋值 0.25
    values(100);                             //没有给可选参数赋值
?>
```

结果为：价格:125　　价格:100



当使用默认参数时，默认参数必须放在非默认参数的右侧，否则函数可能出错。



从 PHP 5 开始，默认值也可以通过引用传递。

3.8.3　从函数中返回值

在 3.8.1 节中介绍了如何定义和调用一个函数，并且讲解了如何在函数间传递值，本节将讲解函数的返回值。通常，函数将返回值传递给调用者的方式是使用关键字 return()。

return()将函数的值返回给函数的调用者，即将程序控制权返回到调用者的作用域。如果在全局作用域内使用 return()关键字，那么将终止脚本的执行。

【例 3.27】　本实例使用 return()函数返回一个操作数。先定义函数 values，函数的作用是输入物品的单价、重量，然后计算总金额，最后输出商品的价格。实例代码如下：**（实例位置：光盘\TM\sl\3\27）**

```
<?php
    function values($price,$tax=0.45){      //定义一个函数，函数中的一个参数有默认值
        $price=$price+($price*$tax);        //计算物品金额
        return $price;                      //返回金额
    }
    echo values(100);                        //调用函数
?>
```

结果为：145

return 语句只能返回一个参数，也即只能返回一个值，不能一次返回多个。如果要返回多个结果，就要在函数中定义一个数组，将返回值存储在数组中返回。

3.8.4 变量函数

PHP 支持变量函数。下面通过一个实例来介绍变量函数的具体应用。

【例 3.28】 本例首先定义 3 个函数，接着声明一个变量，通过变量来访问不同的函数。实例代码如下：**（实例位置：光盘\TM\sl\3\28）**

```
<?php
function come() {                        //定义 come()函数
    echo "来了<p>";
}
function go($name = "jack") {            //定义 go 函数
    echo " $name 走了<p>";
}
function back($string)                   //定义 back 函数
{
    echo "又回来了，$string<p>";
}
$func = "come";                          //声明一个变量，将变量赋值为“come”
$func();                                 //使用变量函数来调用函数 come()
$func = "go";                            //重新给变量赋值
$func("Tom");                            //使用变量函数来调用函数 go()
$func = "back";                          //重新给变量赋值
$func("Lily");                           //使用变量函数来调用函数 back();
?>
```

运行结果如图 3.15 所示。

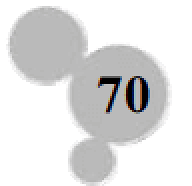


图 3.15 变量函数

可以看到，函数的调用是通过改变变量名来实现的，通过在变量名后面加上一对小括号，PHP 将自动寻找与变量名相同的函数，并且执行它。如果找不到对应的函数，系统将会报错。这个技术可以用于实现回调函数和函数表等。

3.8.5 对函数的引用

在 3.8.2 节中的参数传递中有按引用传递的方式，可以修改实参的内容。引用不仅可用于普通变量、



函数参数，也可作用于函数本身。对函数的引用，就是对函数返回结果的引用。

【例 3.29】 在本例中，首先定义一个函数，这里需在函数名前加“&”符，接着，变量$str 将引用该函数，最后输出该变量$str，实际上就是$tmp 的值。实例代码如下：（**实例位置：光盘\TM\sl\3\29**）

```
<?php
function &example($tmp=0){                      //定义一个函数，加“&”符
      return $tmp;                              //返回参数$tmp
}
$str = &example("看到了");                       //声明一个函数的引用$str;
echo $str."<p>";                                //输出$str
?>
```

结果为：看到了


注意

和参数传递不同，这里必须在两个地方使用“&”符，用来说明返回的是一个引用。

3.8.6 取消引用

当不再需要引用时，可以取消引用。取消引用使用 unset()函数，它只是断开了变量名和变量内容之间的绑定，而不是销毁变量内容。

【例 3.30】本例首先声明一个变量和变量的引用，输出引用后取消引用，再次调用引用和原变量。可以看到，取消引用后对原变量没有任何影响。实例代码如下：（**实例位置：光盘\TM\sl\3\30**）

```
<?php
      $num = 1234;                              //声明一个整型变量
      $math = &$num;                            //声明一个对变量$num 的引用$math
      echo "\$math is:   ".$math."<br>";        //输出引用$math
      unset($math);                             //取消引用$math
      echo "\$math is: ".$math."<br>";          //再次输出引用
      echo "\$num is:   ".$num;                 //输出原变量
?>
```

运行结果如图 3.16 所示。



图 3.16　取消引用

3.9 PHP 编码规范

视频讲解：光盘\TM\lx\3\PHP 编码规范.exe

很多初学者对编码规范很不以为然，认为对程序开发没有什么帮助，甚至因为要遵循规范而影响了学习和开发的进度。或者因为经过一段时间的使用，已经形成了自己的一套风格，所以不愿意去改变。这种想法是很危险的。

举例说明，如今的 Web 开发，不再是一个人就可以全部完成的，尤其是一些大型的项目，要十几人，甚至几十人来共同完成。在开发过程中，难免会有新的开发人员参与进来，那么这个新的开发人员在阅读前任留下的代码时，就会有问题了——这个变量起到什么作用？那个函数实现什么功能？TmpClass 类在哪里被使用到了……诸如此类。这时，编码规范的重要性就体现出来了。

3.9.1 什么是编码规范

以 PHP 开发为例，编码规范就是融合了开发人员长时间积累下来的经验，形成了一种良好统一的编程风格，这种良好统一的编程风格会在团队开发或二次开发时起到事半功倍的效果。编码规范是一种总结性的说明和介绍，并不是强制性的规则。从项目长远的发展以及团队效率来考虑，遵守编码规范是十分必要的。

遵守编码规范的好处如下：

☑ 编码规范是团队开发成员的基本要求。

☑ 开发人员可以了解任何代码，理清程序的状况。

☑ 提高程序的可读性，有利于相关设计人员交流，提高软件质量。

☑ 防止新接触 PHP 的人出于节省时间的需要，自创一套风格并养成终生的习惯。

☑ 有助于程序的维护，降低软件成本。

☑ 有利于团队管理，实现团队后备资源的可重用。

3.9.2 PHP 书写规则

1. 缩进

使用制表符（<Tab>键）缩进，缩进单位为 4 个空格左右。如果开发工具的种类多样，则需要在开发工具中统一设置。

2. 大括号{}

有两种大括号放置规则是可以使用的：

☑ 将大括号放到关键字的下方、同列。



```
if ($expr)
{
    …
}
```

☑　首括号与关键词同行，尾括号与关键字同列。

```
if ($expr){
    …
}
```

两种方式并无太大差别，但多数人都习惯选择第一种方式。

3．关键字、小括号、函数、运算符

☑　不要把小括号和关键字紧贴在一起，要用空格隔开它们。如：

```
if ($expr){                    //if 和“(”之间有一个空格
    …
}
```

☑　小括号和函数要紧贴在一起。以便区分关键字和函数。如：

```
round($num)                    //round 和“(”之间没有空格
```

☑　运算符与两边的变量或表达式要有一个空格（字符连接运算符“.”除外）。如：

```
while ($boo == true){          //$boo 和“==”，true 和“==”之间都有一个空格
    …
}
```

☑　当代码段较大时，上、下应当加入空白行，两个代码块之间只使用一个空行，禁止使用多行。
☑　尽量不要在 return 返回语句中使用小括号。如：

```
return 1;                      //除非是必要，否则不需要使用小括号
```

3.9.3　PHP 命名规则

就一般约定而言，类、函数和变量的名字应该能够让代码阅读者容易地知道这些代码的作用，应该避免使用模棱两可的命名。

1．类命名

☑　使用大写字母作为词的分隔，其他的字母均使用小写。
☑　名字的首字母使用大写。
☑　不要使用下划线('_')。

如：Name、SuperMan、BigClassObject。

2．类属性命名

☑ 属性命名应该以字符“m”为前缀。

☑ 前缀“m”后采用与类命名一致的规则。

☑ “m”总是在名字的开头起修饰作用，就像以“r”开头表示引用一样。

如：mValue、mLongString 等。

3．方法命名

方法的作用都是执行一个动作，达到一个目的。所以名称应该说明方法是做什么。一般名称的前缀和后缀都有一定的规律，如：Is（判断），Get（得到），Set（设置）。

方法的命名规范和类命名是一致的。如：

```
class StartStudy{                   //设置类
      $mLessonOne = "";             //设置类属性
      $mLessonTwo = "";             //设置类属性
      function GetLessonOne(){      //定义方法，得到属性 mLessonOne 的值
            …
      }
}
```

4．方法中参数命名

☑ 第一个字符使用小写字母。

☑ 在首字符后的所有字符都按照类命名规则首字符大写。

如以下代码：

```
class EchoAnyWord{
      function EchoWord($firstWord，$secondWord){
            …
      }
}
```

5．变量命名

☑ 所有字母都使用小写。

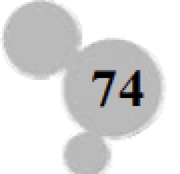
☑ 使用'_'作为每个词的分界。

如：$msg_error、$chk_pwd 等。

6．引用变量

引用变量要带有“r”前缀。如：

```
class Example{
      $mExam   = "";
      function SetExam(&$rExam){
            …
```



```
	}
	function &rGetExam(){
		…
	}
}
```

7. 全局变量

全局变量应该带前缀“g”。如：global = $gTest、global = $g。

8. 常量/全局常量

常量/全局常量，应该全部使用大写字母，单词之间用'_'来分隔。如：

```
define('DEFAULT_NUM_AVE',90);
define('DEFAULT_NUM_SUM',500);
```

9. 静态变量

静态变量应该带前缀“s”。如：

```
static $sStatus = 1;
```

10. 函数命名

所有的名称都使用小写字母，多个单词使用“_”来分割。如：

```
function this_good_idear(){
	…
}
```

以上的各种命名规则，可以组合一起来使用。如：

```
class OtherExample{
	$msValue = "";				//该参数既是类属性，又是静态变量
}
```

说明

这里介绍的只是一些简单的书写和名称规则，如果想了解更多的编码规范，可以参考 Zend_Framework 中文参考手册。

3.10　小　　结

本章主要介绍了 PHP 语言的基础知识，包括数据类型、常量、变量、运算符、表达式和自定义函

数，并详细介绍了各种类型之间的转换、系统预定义的常量、变量、算术优先级和如何使用函数。在最后，又学习了 PHP 编码规范。基础知识是一门语言的核心，希望初学者能静下心来，牢牢掌握本章的知识，这样对以后的学习和发展能起到事半功倍的效果。

3.11 练习与实践

1．动态网页的特点是能够人机交互，但有时却需要限制用户的输入。使用 PHP 函数判断输入（这里先假定一个变量）数据是否符合下列要求：输入必须为全数字、输入字符串长度不允许超过 25，输入不允许为空。注：获取字符串长度函数为 strlen（string）。**（答案位置：光盘\TM\sl\3\31）**

2．获取当前访问者的电脑信息，如 IP、端口号等。**（答案位置：光盘\TM\sl\3\32）**

3．PHP 的输出语句有 echo、print、printf、print_r。尝试使用这 4 个语句输出数据，看它们之间有什么不同。**（答案位置：光盘\TM\sl\3\33）**

第 4 章

流程控制语句

（视频讲解：30 分钟）

学习了 PHP 基础后，相信读者对 PHP 语言的基本运算有了一些了解，那么现在试着计算下面几个问题：输出 10 以内的偶数、计算 100 的阶乘、列举 1000 以内的所有素数。本章就来学习使用 PHP 语言中的流程控制语句解决上述问题。

PHP 的控制流程有两种：条件控制和循环控制。合理使用这些控制结构可以使程序流程清晰、可读性强，从而提高工作效率。

通过阅读本章，您可以：

- 了解 if 语句的使用
- 了解扩展 if 语句的 else、elseif 关键字
- 掌握 switch…case 条件判断语句
- 掌握 while 循环语句
- 掌握 do…while 循环语句
- 掌握 for 循环语句
- 掌握 foreach 循环语句
- 了解 break/continue 关键字

4.1　条件控制语句

视频讲解：光盘\TM\lx\4\条件控制语句.exe

条件控制语句主要有 if、if…else、if…elseif…else 和 switch 4 种。下面分别来了解和使用。

4.1.1　if 语句

几乎所有的语言（包括 PHP）都有 if 语句，它按照条件选择执行不同的代码片段。PHP 的 if 语句的格式为：

```
if (expr)
    statement ;
```

如果表达式 expr 的值为真，那么就顺序执行 statement 语句；否则，就会跳过该条语句，再往下执行。如果需要执行的语句不只一条，那么可以使用“{ }”，在“{ }”中的语句被称为语句组，其格式为：

```
if(expr){
      statement1;
      statement2;
      …
}
```

if 语句的流程控制图如图 4.1 所示。

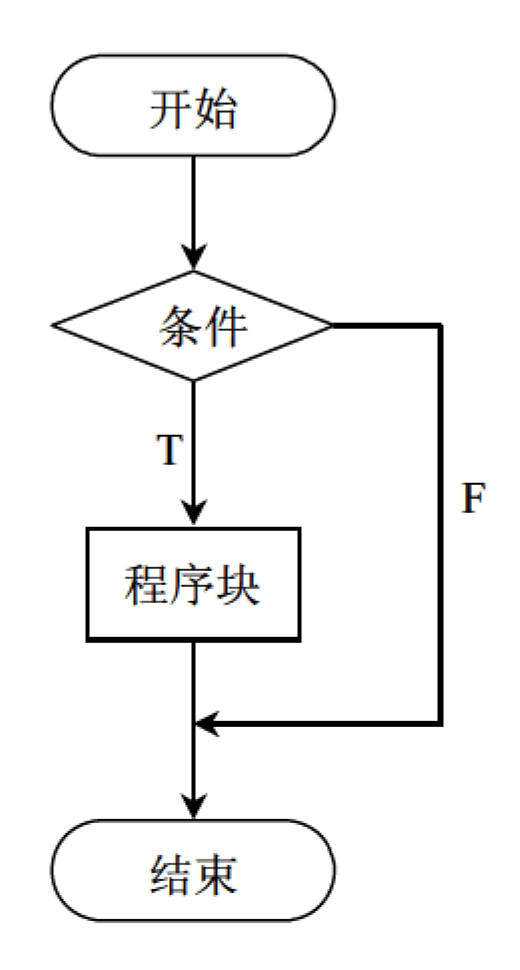
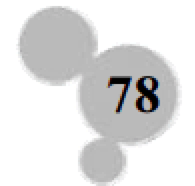


图 4.1　if 语句流程控制图



【例 4.1】 本例首先使用 rand()函数生成一个随机数$num，然后判断这个随机数是不是偶数，如果是，则输出结果。实例代码如下：（**实例位置：光盘\TM\sl\4\1**）

```
<?php
	$num = rand(1,31);				//使用 rand()函数生成一个随机数
	if ($num % 2 == 0){				//判断变量$num 是否为偶数
		echo "\$num = $num";			//如果为偶数，输出表达式和说明文字
		echo "<br>$num 是偶数。";
	}
?>
```

运行结果如图 4.2 所示。



图 4.2　if 语句执行结果



说明

rand()函数的作用是取得一个随机的整数，该函数的格式为：

int rand(int mix, int max)

rand 函数返回 mix～max 之间的一个随机数。如果没有参数，则返回 0～RAND_MAX 之间的随机整数。

4.1.2　if…else 语句

大多时候，总是需要在满足某个条件时执行一条语句，而在不满足该条件时执行其他语句。这时可以使用 else 语句，该语法格式为：

```
if(expr){
	statement1;
}else{
	statement2;
}
```

该语句的含义为：当表达式 expr 为真时，执行 statement1；如果表达式 expr 为假，则执行 statement2。if…else 语句的流程控制图如图 4.3 所示。

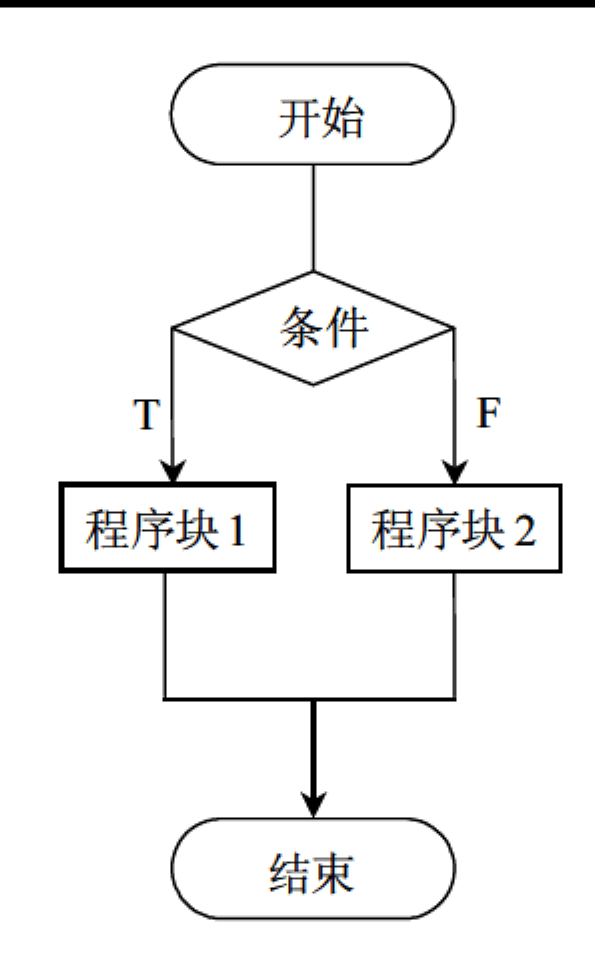


图 4.3　if…else 语句流程控制图

【例 4.2】 本实例以例 4.1 为基础，首先使用 rand()函数生成一个随机数$num，然后判断这个随机数是偶数还是奇数，再根据不同结果显示不同的字符串。实例代码如下：（**实例位置：光盘\TM\sl\4\2**）

```
<?php
	$num = rand(1,31);					//使用 rand()函数生成一个随机数
	if ($num % 2 == 0){					//判断变量$num 是否为偶数
		echo "变量$num 是偶数。";			//如果为偶数
	}else {
		echo "变量$num 为奇数。";			//如果为奇数
	}
?>
```

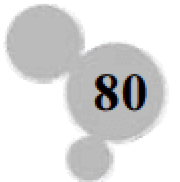
结果为：变量 17 为奇数。

4.1.3　elseif 语句

if…else 语句只能选择两种结果：要么执行真，要么执行假。但有时会出现两种以上的选择，例如：一个班的考试成绩，如果是 90 分以上，则为“优秀”；如果是 60～90 分之间的，则为“良好”；如果低于 60 分，则为“不及格”。这时可以使用 elseif（也可以写作 else if）语句来执行，该语法格式为：

```
if(expr1){
	statement1;
}else if(expr2){
	statement2;
}…
else{
	statementn;
}
```

elseif 语句的流程图如图 4.4 所示。



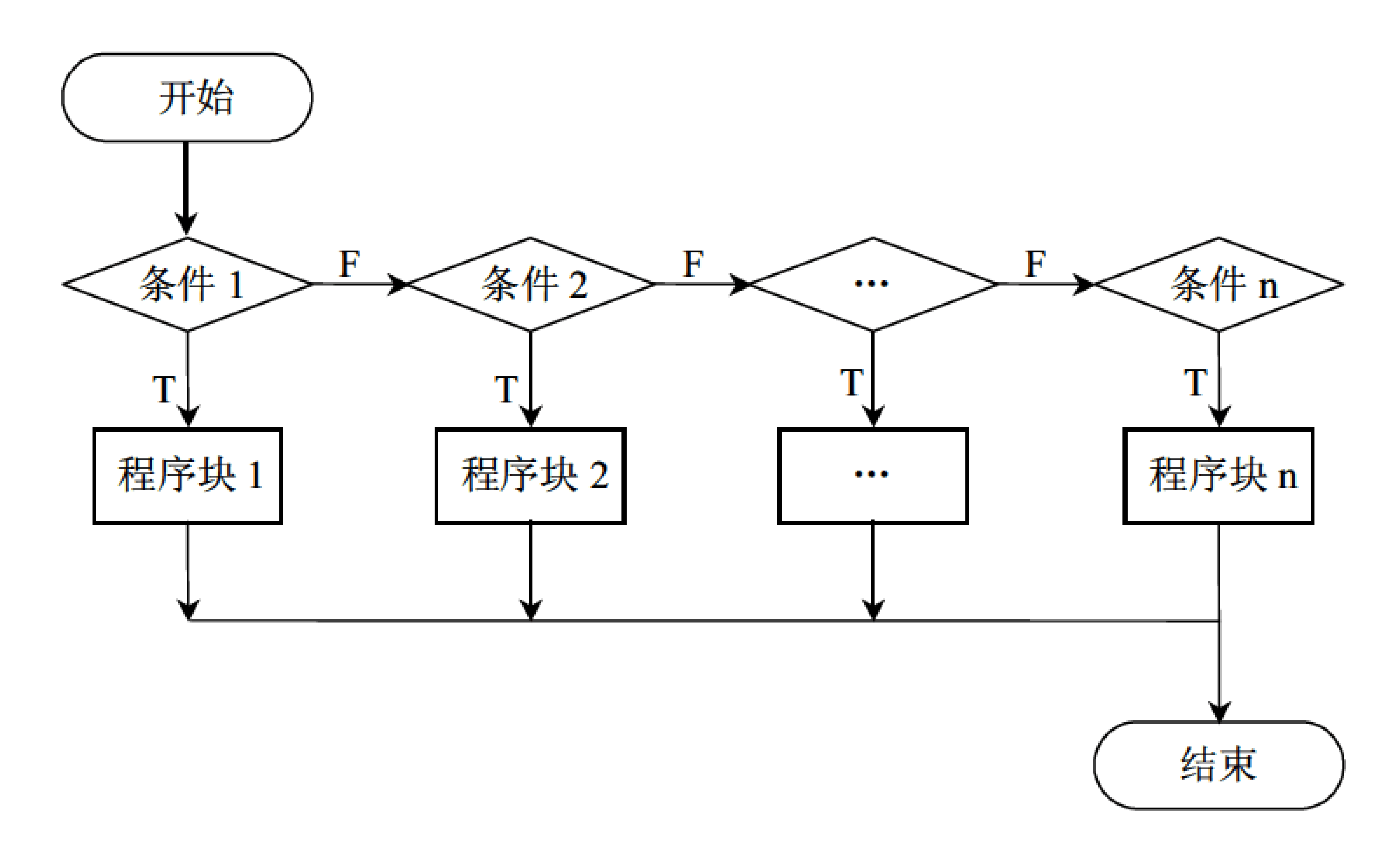


图 4.4　elseif 语句的流程控制图

【例 4.3】 本例通过 elseif 语句，判断今天是这个月的上、中、下旬。实例代码如下：（**实例位置：光盘\TM\sl\4\3**）

```
<?php
	$month = date("n");                                          //设置月份变量$month
	$today = date("j");                                          //设置日期变量$today
	if ($today >= 1 and $today <= 10){                           //判断日期变量是否在 1～10 之间
		echo "今天是".$month."月".$today."日上旬";               //如果是，说明是上旬
	}elseif($today > 10 and $today <= 20){                       //否则判断日期变量是否在 11～20 之间
		echo "今天是".$month."月".$today."日中旬";               //如果是，说明是中旬
	}else{                                                       //如果上面两个判断都不符合要求，则输出默认值
		echo "今天是".$month."月".$today."日下旬";               //说明是本月的下旬
	}
?>
```

结果为：今天是 12 月 7 日上旬

注意

if 语句和 elseif 语句的执行条件是表达式的值为真，而 else 执行条件是表达式的值为假。这里的表达式的值不等于变量的值。如：

```
<?php
$boo = false;
if($boo == false)
	echo "true";
else
	echo "false";
?>
```

该代码段的执行结果为：true

4.1.4 switch…case 多重判断语句

虽然 elseif 语句可以进行多重选择，但使用时十分繁琐。为了避免 if 语句过于冗长，提高程序的可读性，可以使用 switch 分支控制语句。switch 语句的语法格式如下：

```
switch(variable){
	case value1:
		statement1;
		break;
	case value2:
	…
	default:
		default statement;
}
```

switch 语句根据 variable 的值，依次与 case 中的 value 值相比较，如果不相等，继续查找下一个 case；如果相等，就执行对应的语句，直到 switch 语句结束或遇到 break 为止。一般来说，switch 语句最终都有一个默认值 default，如果在前面的 case 中没有找到相符的条件，则输出默认语句，和 else 语句类似。

switch 语句的流程控制图如图 4.5 所示。

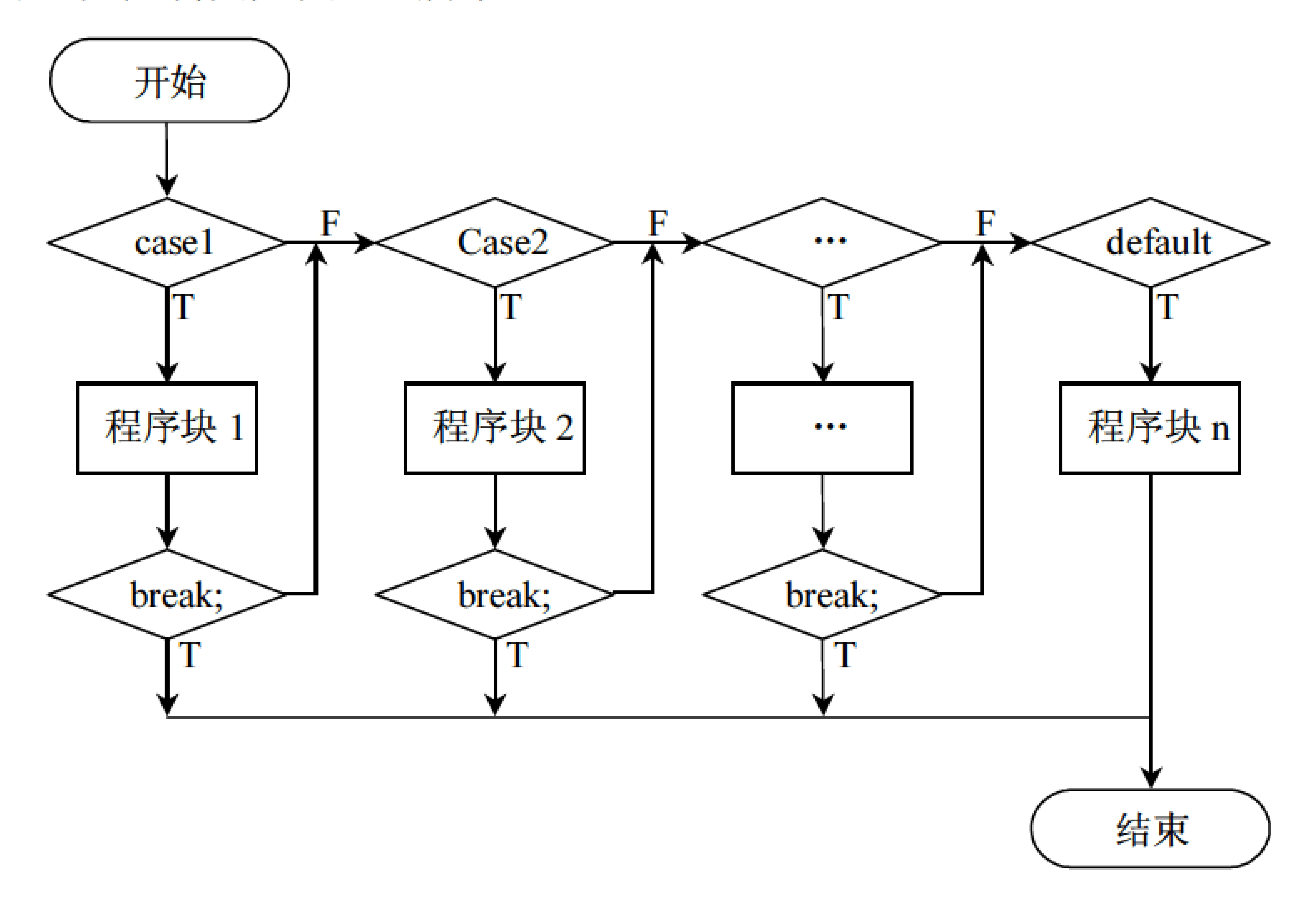
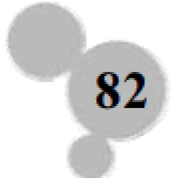


图 4.5　switch 语句流程控制图

【例 4.4】 本实例中应用 switch 语句设计网站的布局，将网站头、尾文件设置为固定不变的板块，导航条也作为固定板块，而在主显示区中，应用 switch 语句根据超链接中传递的值不同，显示不同的内容，实例代码如下：（**实例位置：光盘\TM\sl\4\4**）

```
	<?php
		switch($_GET[lmbs]){			//获取超链接传递的变量
			case "最新商品":			//判断如果变量的值等于“最新商品”
```



```
            include "new.php";          //则执行该语句
        break;                          //否则跳出循环
            case "热门商品":
            include "jollification.php";
        break;
            case "推荐商品":
            include "commend.php";
        break;
            case "订单查询":
            include "order_form.php";
        break;
            default:                    //判断当该值等于空时，执行下面的语句
            include "new.php";
        break;
        }
    ?>
<map name="Map" id="Map">
    <area shape="rect" coords="9,92,65,113" href="#" />
    <area shape="rect" coords="78,89,131,115" href="index.php?lmbs=<?php echo urlencode("最新商品");?>" />
    <area shape="rect" coords="145,92,201,114" href="index.php?lmbs=<?php echo urlencode("推荐商品");?>" />
    <area shape="rect" coords="212,91,268,114" href="index.php?lmbs=<?php echo urlencode("热门商品");?>" />
    <area shape="rect" coords="474,93,529,113" href="index.php?lmbs=<?php echo urlencode("订单查询");?>" />
</map>
```

运行结果如图 4.6 所示。



图 4.6　switch 多重判断语句

注意

switch 语句在执行时，即使遇到符合要求的 case 语句段，也会继续往下执行，直到 switch 语句结束。为了避免这种浪费时间和资源的行为，一定要在每个 case 语句段后加上 break 语句。这里 break 语句的意思是跳出当前循环，在 4.2.6 节中将详细介绍 break 语句。

4.2 循环控制语句

视频讲解：光盘\TM\lx\4\循环控制语句.exe

在 4.1 节中学习了条件判断语句，可以根据条件选择执行不同的语句。但有时需要重复使用某段代码或函数，以本章开始的第二个问题为例，如果要人工输入“1*2*3*4…*100”，无疑是非常繁琐的，但使用循环控制语句就能快速完成计算，下面来学习循环控制语句：while、do…while、for 和 foreach。

4.2.1 while 循环语句

while 循环是 PHP 中最简单的循环语句，它的语法格式为：

```
while (expr){
      statement
}
```

当表达式 expr 的值为真时，将执行 statement 语句，执行结束后，再返回到 expr 表达式继续进行判断。直到表达式的值为假，才跳出循环，执行下面的语句。

while 循环语句的流程控制图如图 4.7 所示。

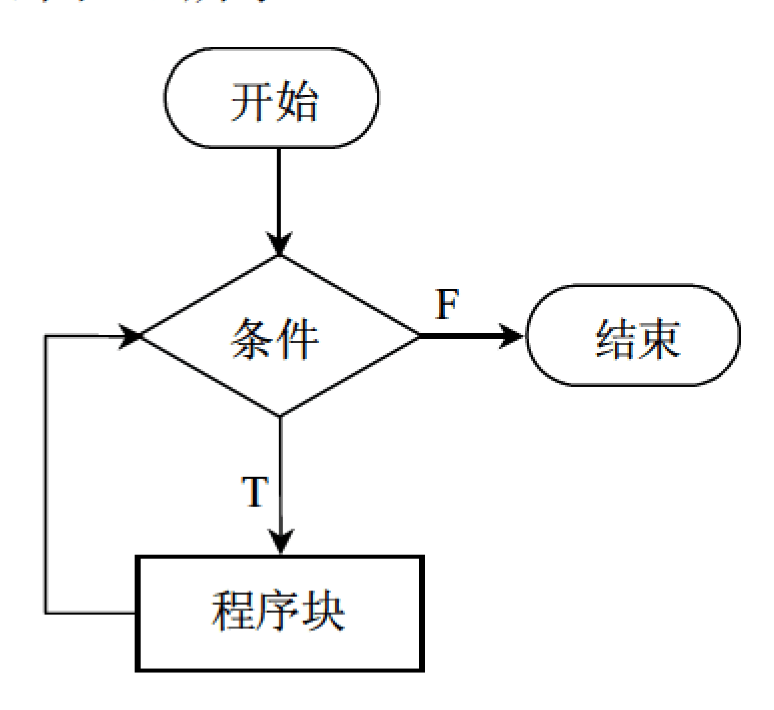
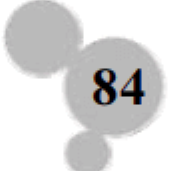


图 4.7 while 语句流程控制图

【例 4.5】 本例将实现 10 以内偶数的输出。从 1～10 依次判断是否为偶数，如果是，则输出；如果不是，则继续下一次循环。实例代码如下：（**实例位置：光盘\TM\sl\4\5**）



```
<?php
	$num = 1;				//声明一个整型变量$num
	$str = "10 以内的偶数为：";		//声明一个字符变量$str
	while($num <= 10){			//判断变量$num 是否小于 10
		if($num % 2 == 0){		//如果小于 10，则判断$num 是否为偶数
			$str .= $num." ";	//如果当前变量为偶数，则添加到字符变量$str 的后面
		}
		$num++;				//变量$num 加 1
	}
	echo $str;				//循环结束后，输出字符串$strt
?>
```

结果为：10 以内的偶数为：2 4 6 8 10

4.2.2　do…while 循环语句

while 语句还有另一种形式的表示，即 do…while。两者的区别在于，do…while 要比 while 语句多循环一次。当 while 表达式的值为假时，while 循环直接跳出当前循环；而 do…while 语句则是先执行一遍程序块，然后再对表达式进行判断。do…while 语句的流程控制图如图 4.8 所示。

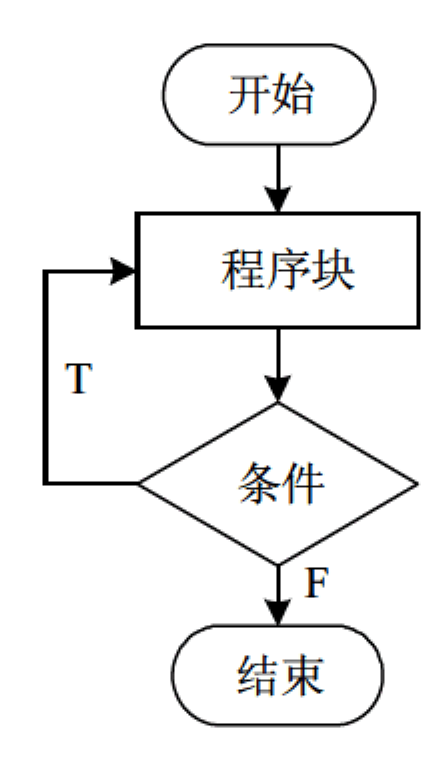


图 4.8　do…while 循环语句流程控制图

【例 4.6】 下面通过两个语句的运行对比来了解两者的不同。实例代码如下：**（实例位置：光盘\TM\sl\4\6）**

```
<?php
	$num = 1;				//声明一个整型变量$num
	while($num != 1){			//使用 while 循环输出
		echo "不会看到。";		//这句话不会输出
	}
	do{					//使用 do…while 循环输出
		echo "会看到。";		//这句话会输出
	}while($num != 1);
?>
```

结果为：会看到。

4.2.3　for 循环语句

for 循环是 PHP 中最复杂的循环结构，它的语法格式为：

```
for (expr1; expr2; expr3){
	statement;
}
```

其中，expr1 在第一次循环时无条件取一次值；expr2 在每次循环开始前求值，如果值为真，则执行 statement，否则跳出循环，继续往下执行；expr3 在每次循环后被执行。for 循环语句的流程控制图如图 4.9 所示。

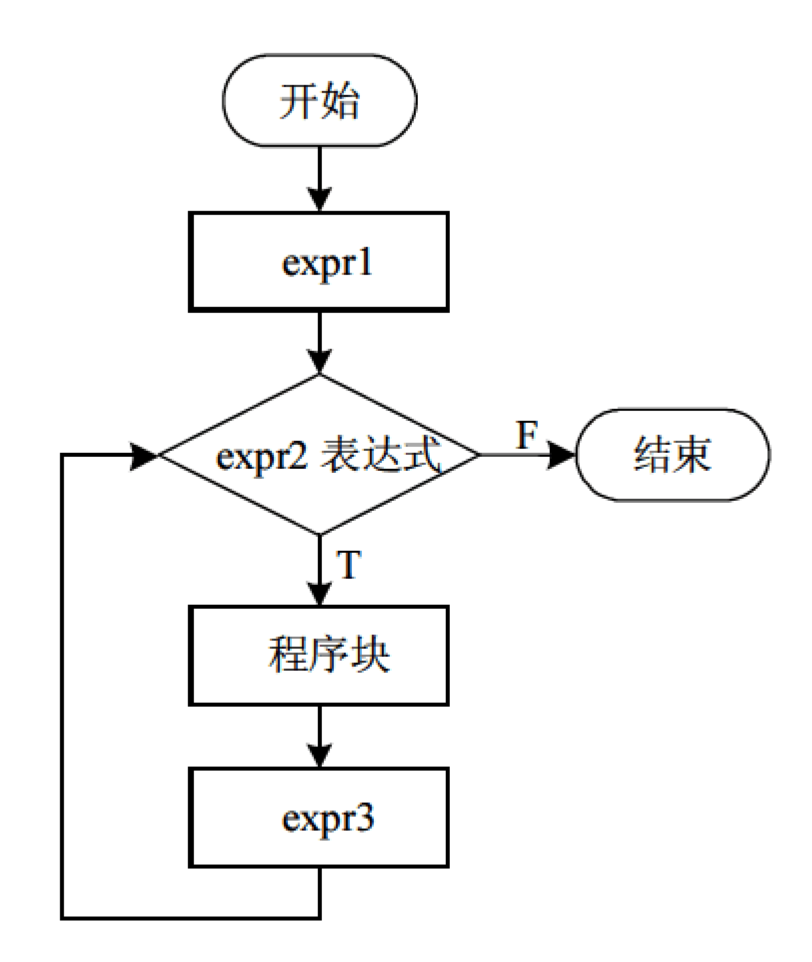


图 4.9　for 循环语句流程控制图

【例 4.7】 下面通过 for 循环来计算 100 的阶乘。实例代码如下：**（实例位置：光盘\TM\sl\4\7）**

```
<?php
	$sum = 1;	//声明整型变量$sum
	for ($i = 1;$i <=100;$i++){
		$sum *= $i;	//当$i 小于 100 时，计算阶乘
	}
	echo "100! = ".$sum;
?>
```

结果为：100! = 9.33262154439E+157

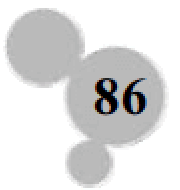
注意

在 for 语句中无论采用循环变量递增或递减的方式，前提是一定要保证循环能够结束，无期限的循环（死循环）将导致程序的崩溃。

4.2.4　foreach 循环语句

foreach 循环是 PHP 4 引进来的，只能用于数组。在 PHP 5 中，又增加了对对象的支持。该语句的语法格式为：

```
foreach (array_expression as $value)
    statement
```



或

```
foreach (array_expression as $key => $value)
      statement
```

foreach 语句将遍历数组 array_expression，每次循环时，将当前数组中的值赋给$value（或是$key 和$value），同时，数组指针向后移动直到遍历结束。当使用 foreach 语句时，数组指针将自动被重置，所以不需要手动设置指针位置。

【例 4.8】 在本例中，应用 foreach 语句输出数组中存储的商品信息。实例代码如下：（**实例位置：光盘\TM\sl\4\8**）

```
<?php
$name = array("1"=>"智能机器人","2"=>"数码相机","3"=>"天翼 3G 手机","4"=>"瑞士手表");
$price = array("1"=>"14998 元","2"=>"2588 元","3"=>"2666 元","4"=>"66698 元");
$counts = array("1"=>1,"2"=>1,"3"=>2,"4"=>1);
echo '<table width="580" border="1" cellpadding="1" cellspacing="1" bordercolor="#FFFFFF" bgcolor="#c17e50">
            <tr>
              <td width="145" align="center" bgcolor="#FFFFFF"   class="STYLE1">商品名称</td>
              <td width="145" align="center" bgcolor="#FFFFFF"   class="STYLE1">价  格</td>
              <td width="145" align="center" bgcolor="#FFFFFF"   class="STYLE1">数量</td>
              <td width="145" align="center" bgcolor="#FFFFFF"   class="STYLE1">金额</td>
 </tr>';
foreach($name as $key=>$value){            //以 book 数组做循环，输出键和值
      echo '<tr>
              <td height="25" align="center" bgcolor="#FFFFFF" class="STYLE2">'.$value.'</td>
              <td align="center" bgcolor="#FFFFFF" class="STYLE2">'.$price[$key].'</td>
              <td align="center" bgcolor="#FFFFFF" class="STYLE2">'.$counts[$key].'</td>
              <td align="center" bgcolor="#FFFFFF" class="STYLE2">'.$counts[$key]*$price[$key].'</td>
</tr>';
}
echo '</table>';
?>
```

运行结果如图 4.10 所示。



图 4.10　使用 foreach 语句输出数组

注意

当试图使用 foreach 语句用于其他数据类型或者未初始化的变量时会产生错误。为了避免这个问题，最好使用 is_array()函数先来判断变量是否为数组类型。如果是，再进行其他操作。

4.2.5 流程控制的另一种书写格式

在一个复杂的 PHP 页面中，可能包含了多个条件语句、循环语句和函数，仅查找匹配的大括号“{}”就非常麻烦。为此，PHP 提供了另一种书写格式，包括 if、while、for、foreach 和 switch 都可以使用。该书写格式的基本形式是：使用冒号“:”来代替左边的大括号“{”；使用 endif;、endwhile;、endfor;、endreach;和 endswitch;来代替右边的大括号“}”。

【例 4.9】 下面使用流程控制实现本节开始时所说的第 3 个问题：列举 1000 以内的所有素数。实例代码如下：**（实例位置：光盘\TM\sl\4\9）**

```
<?php
	$ss = 2;						//声明变量$ss，赋初值为最小的素数
	$max = 1000;					//声明变量$max，赋初值为最大的范围
	$arr = array();					//声明一个数组$arr
	echo $max."以内的素数为：";
	while($ss < $max):				//判断变量是否在允许的范围内
		$boo = false;				//声明一个布尔变量$boo，初值为 false
		foreach($arr as $value):		//使用 foreach 语句遍历$arr 数组
			if($ss % $value == 0):	//如果变量$ss 能够被数组元素整除
				$boo = true;		//将布尔变量赋值为 true
				break;			//跳出当前循环
			endif;
		endforeach;
		if(!$boo):				//判断变量$boo 值
			echo $ss." ";			//如果$boo 为假，则说明当前变量$ss 为素数，输出素数
			$arr[count($arr)] = $ss;	//同时存到数组中
		endif;
		$ss++;					//变量$ss 加 1
	endwhile;					//结束循环
?>
```

运行结果如图 4.11 所示。

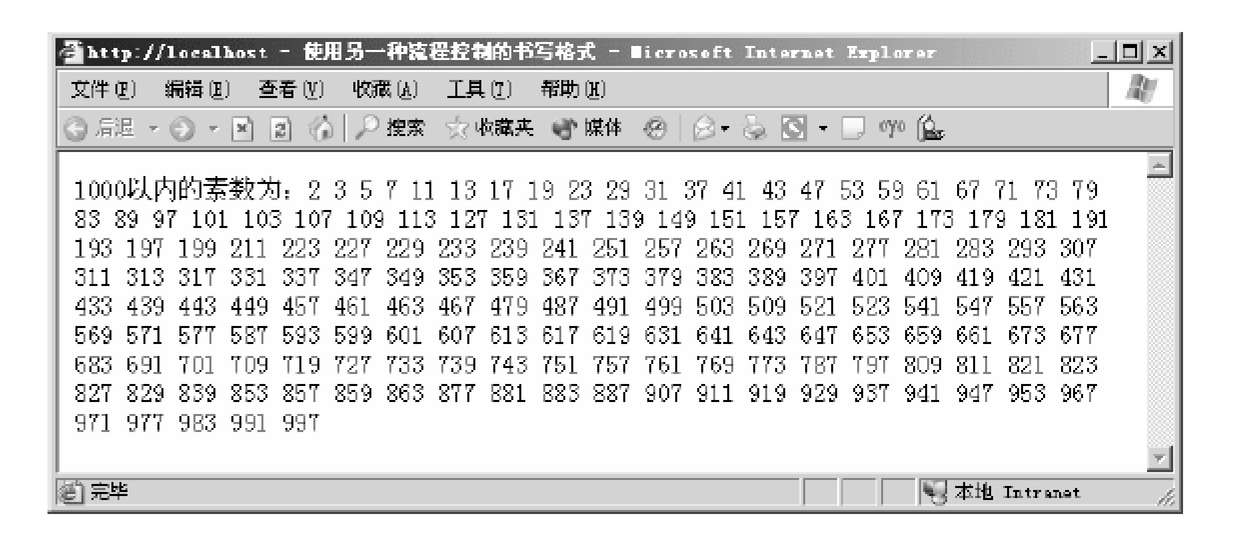
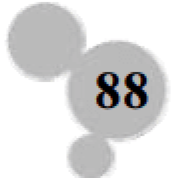


图 4.11　使用另一种流程控制的书写格式



4.2.6　使用 break/continue 语句跳出循环

视频讲解：光盘\TM\lx\4\结束循环语句.exe

在使用循环语句时，有时不确定循环的次数，遇到这样的情况可以使用无限循环，如：

```
while(true){
    …
}
```

或

```
for(;;){
    …
}
```

只有当程序块满足一定条件后才跳出循环，跳出循环使用的关键字是 break 和 continue。

1. break

break 关键字可以终止当前的循环，包括 while、do…while、for、foreach 和 switch 在内的所有控制语句。下面来看一个实例。

【例 4.10】 本例将使用一个 while 循环，while 后面的判断式的值为 true，即为一个无限循环。在 while 程序块中将声明一个随机数变量$tmp，只有当生成的随机数等于 10 时，使用 break 语句跳出循环。实例代码如下：（**实例位置：光盘\TM\sl\4\10**）

```
<?php
while(true){                              //使用 while 循环
    $tmp = rand(1,20);                    //声明一个随机数变量$tmp
    echo $tmp." ";                        //输出随机数
    if($tmp == 10){                       //判断随机数是否等于 10
        echo "<p>变量等于 10，终止循环";
        break;                            //如果等于 10，使用 break 语句跳出循环
    }
}
?>
```

运行结果如图 4.12 所示。

break 语句不仅可以跳出当前的循环，还可以指定跳出几重循环。格式为：

```
break $num;
```

参数$num 指定要跳出几层循环。break 关键字的流程控制图如图 4.13 所示。



图 4.12　使用 break 语句跳出循环

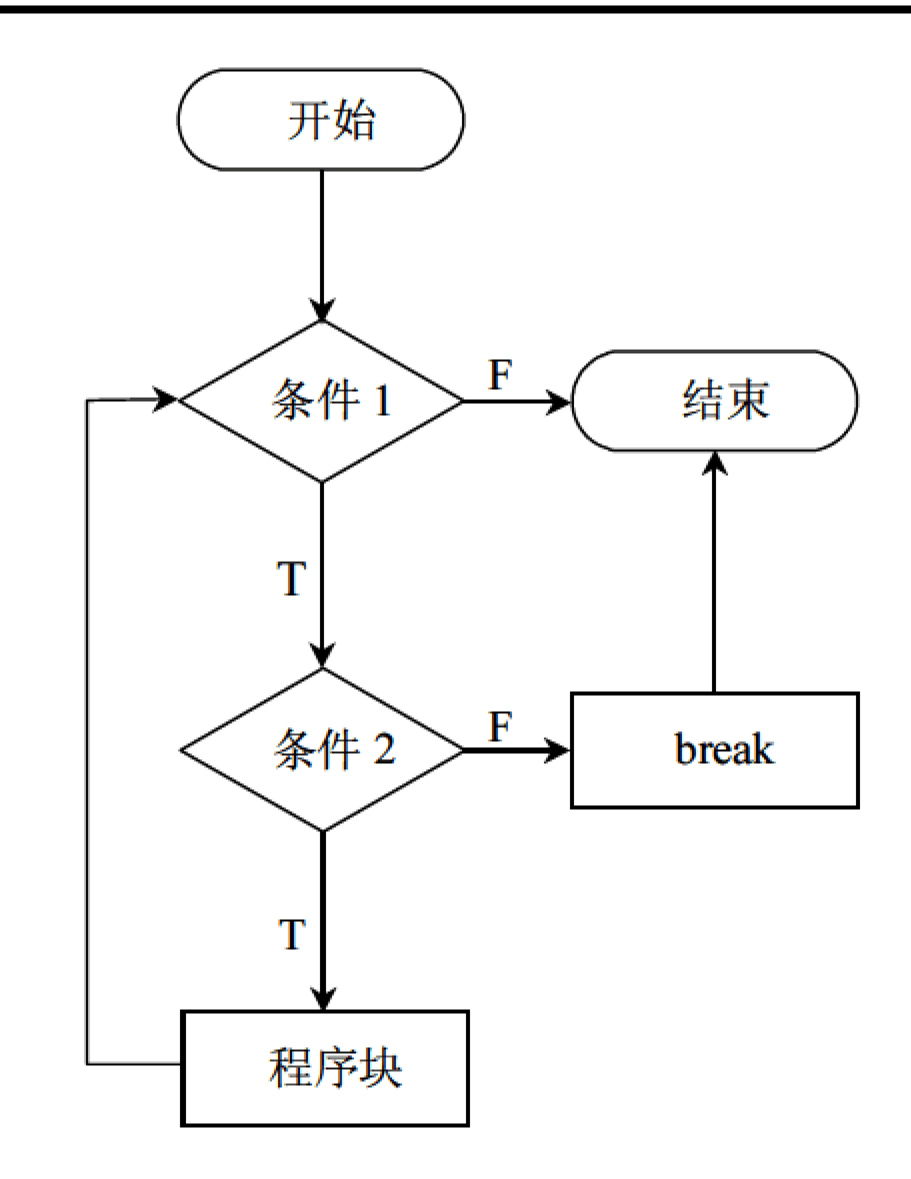
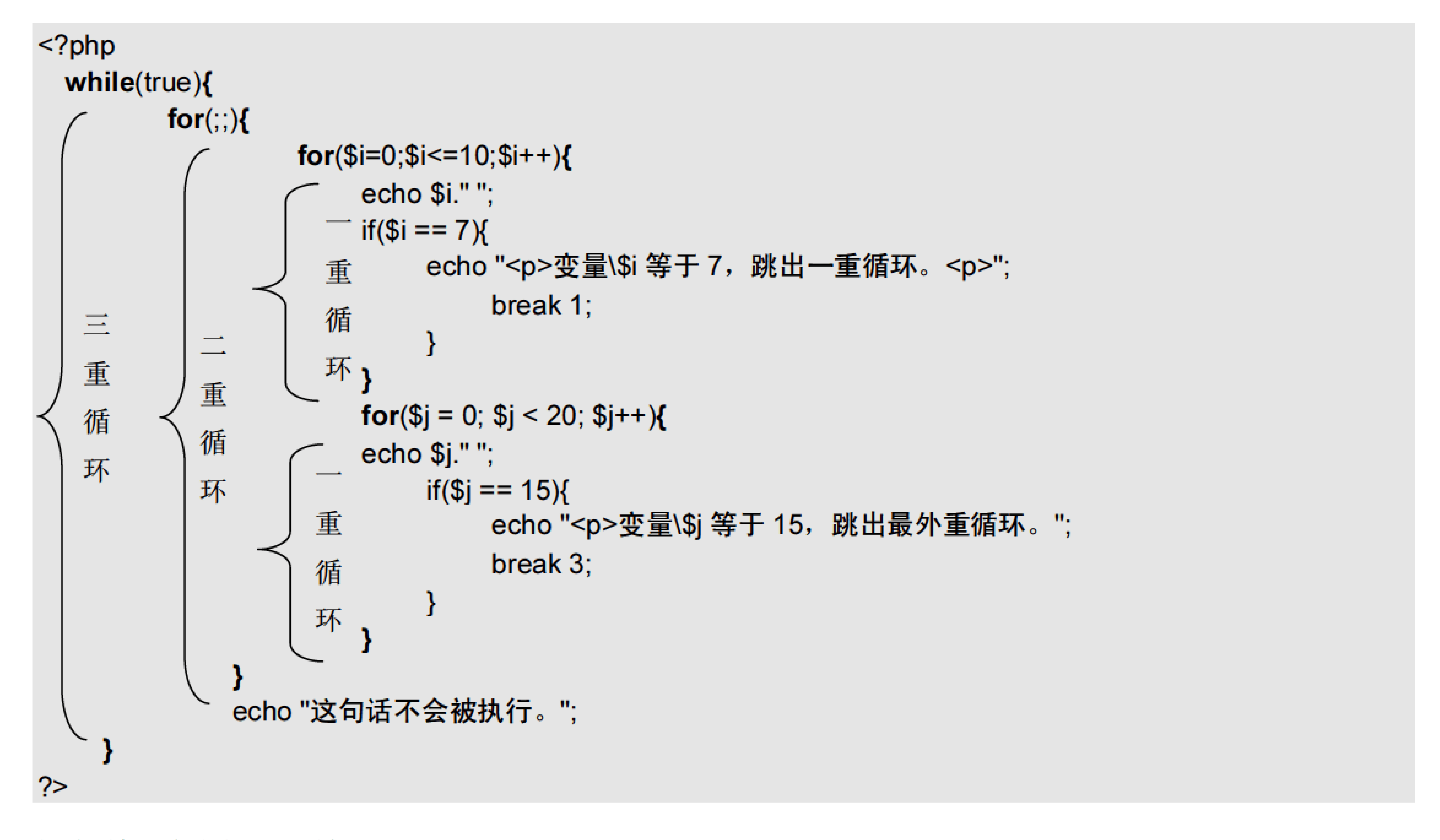


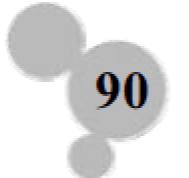
图 4.13　break 流程控制图

【例 4.11】 本例共有 3 层循环，最外层的 while 循环和中间层的 for 循环是无限循环，最里面并列两个 for 循环：程序首先执行第一个 for 循环，当变量$i 等于 7 时，跳出当前循环（一重循环），继续执行第二个 for 循环，当第二个 for 循环中的变量$j 等于 15 时，将直接跳出最外层循环。实例代码如下：**（实例位置：光盘\TM\sl\4\11）**




```
<?php
  while(true){
        for(;;){
              for($i=0;$i<=10;$i++){
                  echo $i." ";
                  if($i == 7){
                      echo "<p>变量\$i 等于 7，跳出一重循环。<p>";
                          break 1;
                      }
                  }
                  for($j = 0; $j < 20; $j++){
                  echo $j." ";
                      if($j == 15){
                          echo "<p>变量\$j 等于 15，跳出最外重循环。";
                          break 3;
                      }
                  }
        }
        echo "这句话不会被执行。";
  }
?>
```

运行结果如图 4.14 所示。



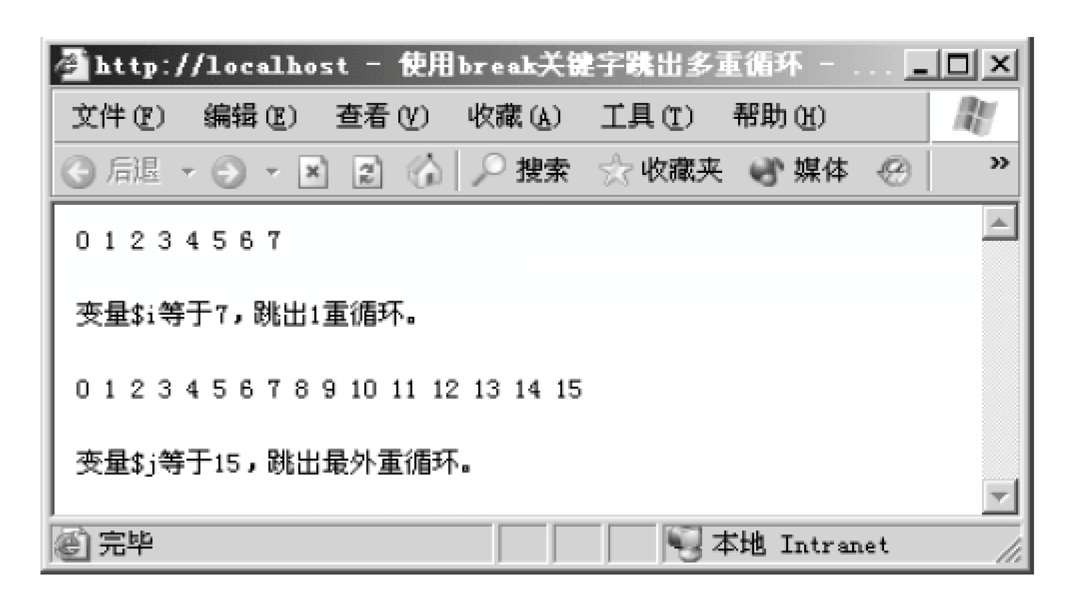


图 4.14　使用 break 关键字跳出多重循环

2．continue

continue 关键字的作用没有 break 强大，continue 只能终止本次循环而进入到下一次循环中，continue 也可以指定跳出几重循环。continue 关键字的流程控制图如图 4.15 所示。

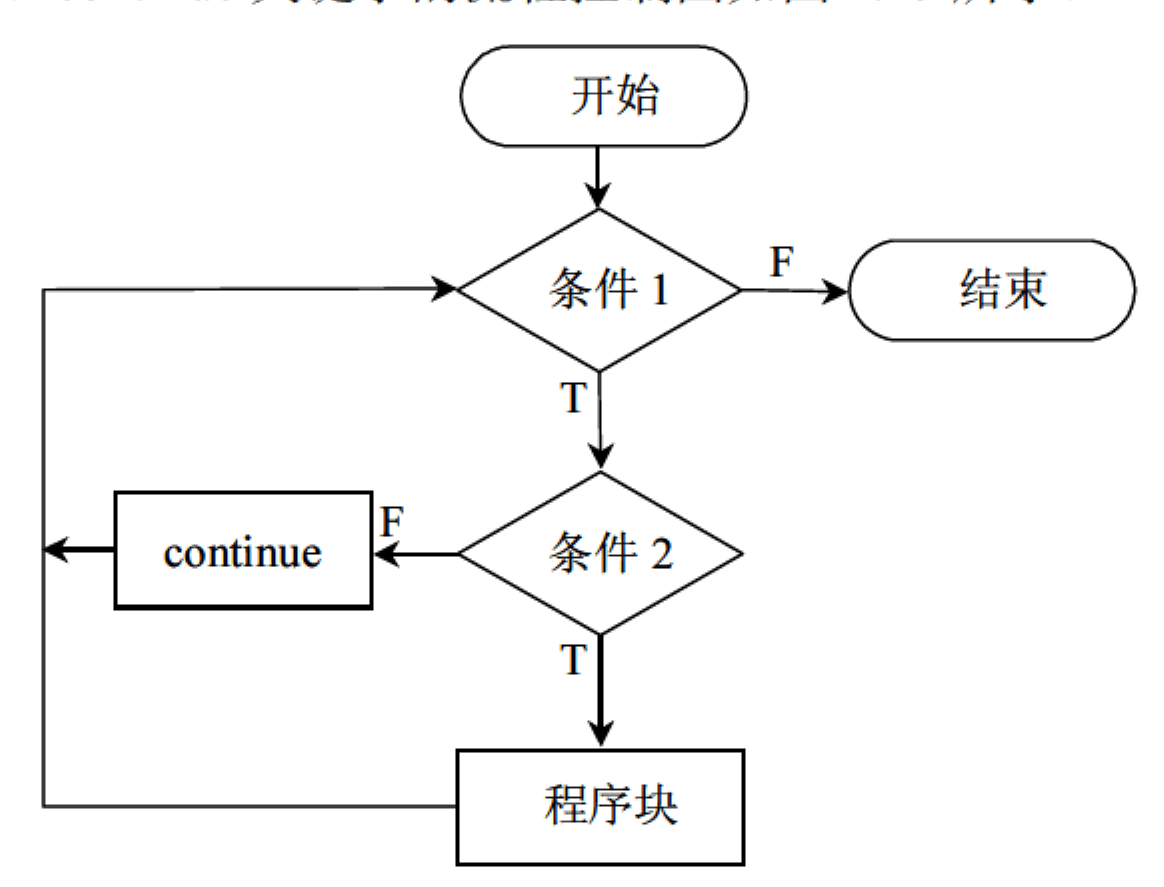


图 4.15　continue 流程控制图

【例 4.12】 本例使用 for 循环输出 A～J 的数组变量。如果变量的数组下标为偶数，则只输出一个空行；如果是奇数，则继续输出。在最里面的循环中，判断当前数组下标是否等于$i，如果不相等，则输出数组变量，否则跳到最外重循环。实例代码如下：**（实例位置：光盘\TM\sl\4\12）**

```
<?php
	$arr = array("A","B","C","D","E","F","G","H","I","J");		//声明一个数组变量$arr
	for($i = 0; $i < 10; $i++){						//使用 for 循环
		echo "<br>";
		if($i % 2 == 0){							//如果$i 的值为偶数，则跳出本次循环
			continue;
		}
		for(;;){								//无限循环
			for($j = 0; $j < count($arr); $j++){		//再次使用 for 循环输出数组变量
				if($j == $i){					//如果当前输出的数组下标等于$i
					continue 3;				//跳出最外重循环
				}else{
					echo "\$arr[".$j."]=".$arr[$j]." ";	//输出表达式
				}
```

```
                }
            }
            echo "这句话永远不会输出";
        }
?>
```

运行结果如图 4.16 所示。

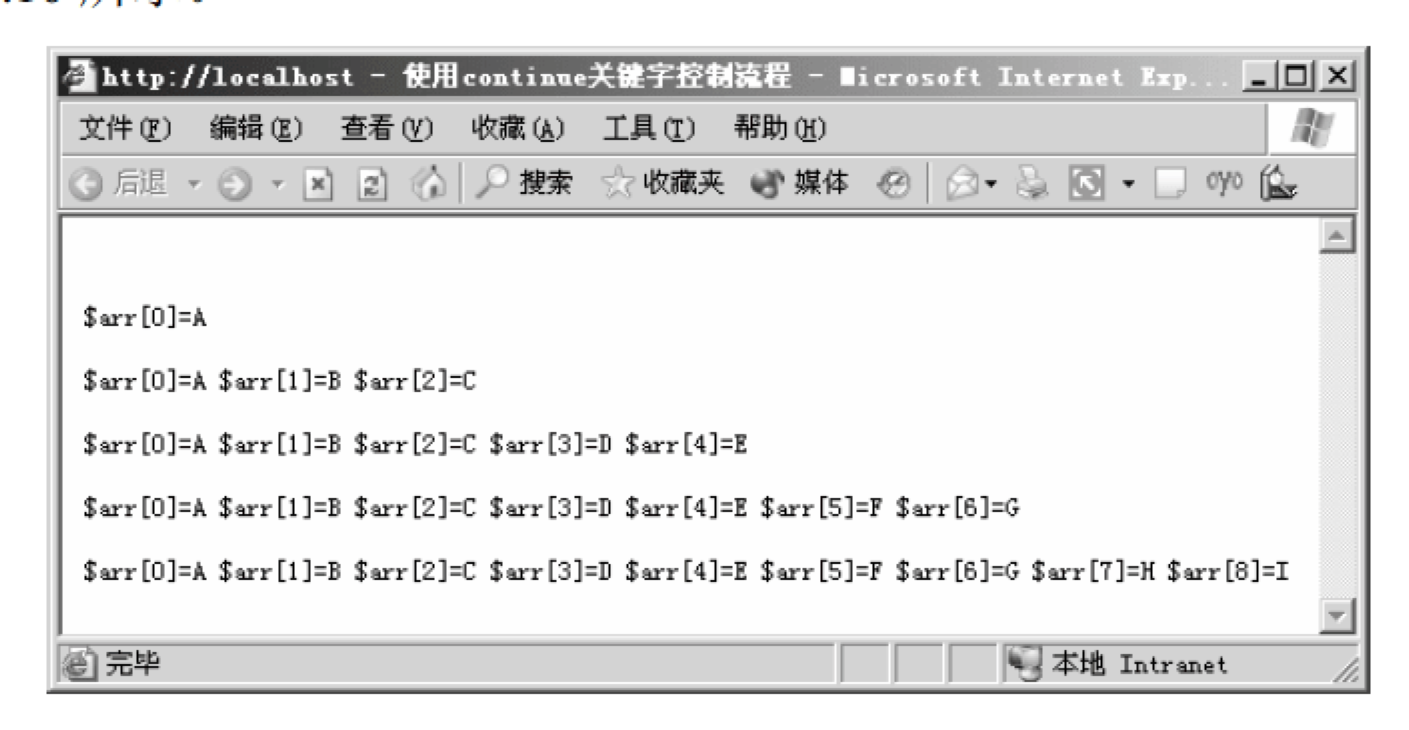


图 4.16　使用 continue 关键字控制流程

4.3　小　　结

本章通过几个简单的数学题学习了 PHP 的流程控制语句。流程控制语句是程序中必不可少的，也是变化最丰富的技术。无论是入门的数学公式，还是高级的复杂算法，都是通过这几个简单的语句来实现的。相信读者学习完本章之后，通过不断地练习和总结，能够掌握一套自己的方法和技巧。

4.4　练习与实践

1．使用循环语句，输出任意一个二维数组。**（答案位置：光盘\TM\sl\4\13）**

2．使用循环控制语句，输出杨辉三角。**（答案位置：光盘\TM\sl\4\14）**

3．使用 while 循环和预定义变量，获取多个参数。参数的个数未定，如：http://localhost/1.php?name=tm&password=111&date=20080424&id=1…。**（答案位置：光盘\TM\sl\4\15）**

第 5 章

字符串操作

（视频讲解：1 小时 16 分钟）

在 Web 编程中，字符串总是会被大量地生成和处理。正确地使用和处理字符串，对于 PHP 程序员来说越来越重要。本章从最简单的字符串定义一直引导读者到高层字符串处理技巧，希望广大读者能够通过本章的学习，了解和掌握 PHP 字符串，达到举一反三的目的，为了解和学习其他的字符串处理技术奠定良好的基础。

通过阅读本章，您可以：

- 了解字符串
- 掌握单引号和双引号的使用方法及区别
- 掌握字符串的连接方法
- 熟悉去除字符串中的空格
- 熟悉字符串的转义及还原
- 掌握获取字符串长度的方法
- 掌握连接和分割字符串的方法
- 熟练掌握字符串的定位技术
- 掌握添加、删除和截取字符串技术
- 掌握查找和替换字符串技术
- 了解字符串的格式化技术

5.1 字符串简介

视频讲解：光盘\TM\lx\5\了解字符串.exe

字符串是指由零个或多个字符构成的一个集合，这里所说的字符主要包含以下几种类型：

☑ 数字类型，如 1、2、3 等。

☑ 字母类型，如 a、b、c、d 等。

☑ 特殊字符，如#、$、%、^、&等。

☑ 不可见字符，如\n（换行符）、\r（回车符）、\t（Tab 字符）等。

其中，不可见字符是比较特殊的一组字符，它用来控制字符串格式化输出，在浏览器上不可见，只能看到字符串输出的结果。

例如：

```
<?php
echo  "fruit\rpear\napple\tbanana";                    //输出字符串
?>
```

结果为：

```
fruit
pear
apple      banana
```



技巧

本实例的运行结果在 IE 浏览器上不可见，需要在 IE 浏览器中选择“查看”/“源文件”命令来查看字符串的输出结果。

5.2 单引号和双引号的区别

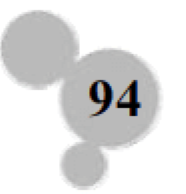
视频讲解：光盘\TM\lx\5\单引号和双引号的区别.exe

字符串通常以串的整体作为操作对象，一般用双引号或者单引号标识一个字符串。单引号和双引号在使用上有一定区别。

下面分别使用双引号和单引号来定义一个字符串。

例如：

```
<?php
$str1 = "I Like PHP";                                   //使用双引号定义一个字符串
```



```
$str2 = 'I Like PHP';                          //使用单引号定义一个字符串
echo $str1;                                    //输出双引号中的字符串
echo $str2;                                    //输出单引号中的字符串
?>
```

结果为：

```
I Like PHP
I Like PHP
```

从上面的结果中可以看出，对于定义的普通字符串看不出两者之间的区别。而通过对变量的处理，即可轻松地理解两者之间的区别。例如：

```
<?php
$test = "PHP";
$str = "I Like  $test";
$str1 = 'I Like $test';
echo $str;                                     //输出双引号中的字符串
echo $str1;                                    //输出单引号中的字符串
?>
```

结果为：

```
I Like PHP
I Like $test
```

从以上代码中可以看出，双引号中的内容是经过 PHP 的语法分析器解析过的，任何变量在双引号中都会被转换为它的值进行输出显示；而单引号的内容是"所见即所得"的，无论有无变量，都被当作普通字符串进行原样输出。

技巧

单引号串和双引号串在 PHP 中的处理是不相同的。双引号串中的内容可以被解释并替换，而单引号串中的内容则只能被作为普通字符进行处理。

注意

在进行 SQL 查询之前，所有字符串都必须加单引号，以避免可能的注入漏洞和 SQL 错误。

5.3　字符串的连接符

视频讲解：光盘\TM\lx\5\字符串的连接符.exe

半角句号"."是字符串连接符，可以把两个或两个以上的字符串连接成一个字符串。

例如：

```
<?php
$name="明日编程词典：";
$url = "www.mrbccd;
echo $name. $url.".com";
?>
```

结果为：明日编程词典：www.mrbccd.com

应用字符串连接符号无法实现大量简单字符串的连接，PHP 允许程序员在双引号中直接包含字符串变量，当 echo 语句后面使用的是双引号（"）时，可以使用下面的格式来达到同样的效果。

例如：

```
<?php
$name="明日编程词典:";
$url = "www.mrbccd";
echo "$name$url.com";                    //双引号里的变量同一般的字符串自动进行区分
?>
```

结果为：明日编程词典：www.mrbccd.com

5.4　字符串操作

视频讲解：光盘\TM\lx\5\字符串操作.exe

字符串的操作在 PHP 编程中占有重要的地位，几乎所有 PHP 脚本的输入与输出都用到字符串。尤其是在 PHP 项目开发过程中，为了实现某项功能，经常需要对某些字符串进行特殊处理，如获取字符串的长度、截取字符串、替换字符串等。在本节中将对 PHP 常用的字符串操作技术进行详细的讲解，并通过具体的实例加深对字符串函数的理解。

5.4.1　去除字符串首尾空格和特殊字符

用户在输入数据时，经常会在无意中输入多余的空格，在有些情况下，字符串中不允许出现空格和特殊字符，此时就需要去除字符串中的空格和特殊字符。在 PHP 中提供了 trim()函数去除字符串左右两边的空格和特殊字符、ltrim()函数去除字符串左边的空格和特殊字符、rtrim()函数去除字符串中右边的空格和特殊字符。

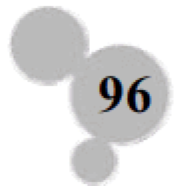
1. trim()函数

trim()函数用于去除字符串开始位置以及结束位置的空格，并返回去掉空格后的字符串。

语法格式如下：

```
string trim(string str [,string charlist]);
```

trim()函数的参数 str 是要操作的字符串对象，参数 charlist 为可选参数，指定需要从指定的字符串



中删除哪些字符，如果不设置该参数，则所有的可选字符都将被删除。trim()函数的参数 charlist 的可选值如表 5.1 所示。

表 5.1　trim()函数的参数 charlist 的可选值

参　数　值	说　　明
\0	NULL，空值
\t	tab，制表符
\n	换行符
\x0B	垂直制表符
\r	回车符
" "	空格

注意

除了以上默认的过滤字符列表外，也可以在 charlist 参数中提供要过滤的特殊字符。

【例 5.1】 使用 trim()函数去除字符串左右两边的空格及特殊字符“\r\r(: :)”，实例代码如下：**（实例位置：光盘\TM\sl\5\1）**

```
<?php
$str="\r\r(:@_@ 创图书编撰伟业 展软件开发雄风 @_@:)          ";
echo trim($str);             //去除字符串左右两边的空格
echo "<br>";                 //执行换行
echo trim($str,"\r\r(: :)"); //去除字符串左右两边的特殊字符\r\r(: :)
?>
```

结果为：

```
(:@_@ 创图书编撰伟业 展软件开发雄风 @_@:)
@_@ 创图书编撰伟业 展软件开发雄风 @_@
```

2．ltrim()函数

ltrim()函数用于去除字符串左边的空格或者指定字符串。

语法格式如下：

```
string ltrim( string str [,string charlist]);
```

【例 5.2】 使用 ltrim()函数去除字符串左边的空格及特殊字符“(:@_@”，实例代码如下：**（实例位置：光盘\TM\sl\5\2）**

```
<?php
$str="    (:@_@   创图书编撰伟业   @_@:)     ";
echo ltrim($str);                //去除字符串左边的空格
echo "<br>";                     //执行换行
echo ltrim($str," (:@_@ ");      //去除字符串左边的特殊字符@_@:)
?>
```

结果为：

```
(:@_@ 创图书编撰伟业 @_@:)
创图书编撰伟业 @_@:)
```

3．rtrim()函数

rtrim()函数用于去除字符串右边的空格。

语法格式如下：

```
String rtrim(string str [,string charlist]);
```

【例 5.3】 使用 rtrim()函数去除字符串右边的空格及特殊字符“@_@:)”，实例代码如下：**（实例位置：光盘\TM\sl\5\3）**

```
<?php
$str="   (:@_@   展软件开发雄风   @_@:)     ";
echo rtrim($str);                    //去除字符串右边的空格
echo "<br>";                         //执行换行
echo rtrim($str," @_@:)");           //去除字符串右边的特殊字符@_@:)
?>
```

结果为：

```
(:@_@展软件开发雄风 @_@:)
(:@_@展软件开发雄风
```

5.4.2　转义、还原字符串数据

字符串转义、还原的方法有两种：一种是手动转义、还原字符串数据，另一种是自动转义、还原字符串数据。下面分别对这两种方法进行详细讲解。

1．手动转义、还原字符串数据

字符串可以用单引号（'）、双引号（""）、定界符（{}）3 种方法定义。而指定一个简单字符串的最简单的方法是用单引号（'）括起来。当使用字符串时，很可能在该串中存在这几种符号与 PHP 脚本混淆的字符，因此必须要做转义语句。这就要在它的前面使用转义符号“\”。

“\”是一个转义符，紧跟在“\”后面的第一个字符将变得没有意义或有特殊意义。如'是字符串的定界符，写为\'时就失去了定界符的意义，变为了普通的单引号'。读者可以通过 echo '\'';输出一个单引号'，同时转义字符“\”也不会显示。

技巧

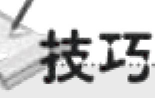
如果要在字符串中表示单引号，则需要用反斜线（\）进行转义。例如，要表示字符串“I'm”，则需要写成“I\'m”。



【例 5.4】 使用转义字符“\”对字符串进行转义，实例代码如下：（**实例位置：光盘\TM\sl\5\4**）

```
<?php
echo ' select * from tb_book where bookname = \'PHP5 从入门到精通\' ';
?>
```

结果为：select * from tb_book where bookname = 'PHP5 从入门到精通'



技巧

对于简单的字符串建议采用手动方法进行字符串转义，而对于数据量较大的字符串，建议采用自动转义函数实现字符串的转义。

说明

手动转义字符串可应用 addcslashes()函数进行字符串还原，其具体的实现方法将在下面进行介绍。

2．自动转义、还原字符串数据

自动转义、还原字符串数据可以应用 PHP 提供的 addslashes()函数和 stripslashes()函数实现。

☑ addslashes()函数

addslashes()函数用来为字符串 str 加入斜线“\”。

语法格式如下：

```
string addslashes ( string str)
```

☑ stripslashes()函数

stripslashes()函数用来将使用 addslashes()函数转义后的字符串 str 返回原样。

语法格式如下：

```
string stripslashes(string str);
```

【例 5.5】 使用自动转义字符 addslashes()函数对字符串进行转义，然后使用 stripslashes()函数进行还原，实例代码如下：（**实例位置：光盘\TM\sl\5\5**）

```
<?php
 $str = "select * from tb_book where bookname = 'PHP5 从入门到精通'";
 echo $str."<br>";                                   //输出字符串
 $a = addslashes($str);                              //对字符串中的特殊字符进行转义
 echo $a."<br>";                                     //输出转义后的字符
 $b = stripslashes($a);                              //对转义后的字符进行还原
 echo $b."<br>";                                     //将字符原义输出
?>
```

运行结果如图 5.1 所示。

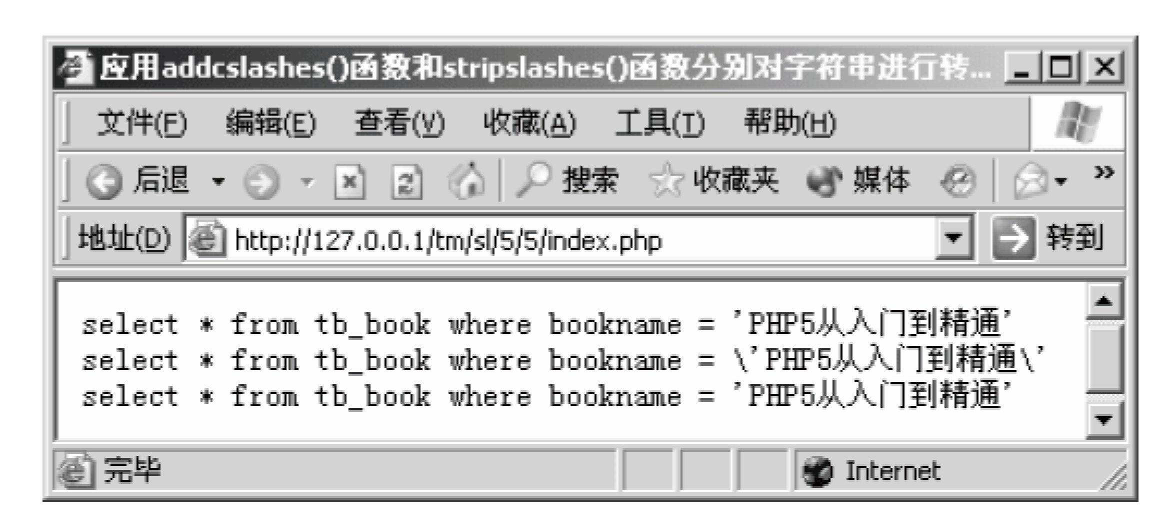


图 5.1　使用 addslashes()函数和 stripslashes()函数分别对字符串进行转义和还原

技巧

所有数据在插入数据库之前，有必要应用 addslashes()函数进行字符串转义，以免特殊字符未经转义在插入数据库时出现错误。另外，对于使用 addslashes()函数实现的自动转义字符串可以使用 stripcslashes()函数进行还原，但数据在插入数据库之前必须再次进行转义。

以上两个函数实现了对指定字符串进行自动转义和还原。除了上面介绍的方法外，还可以对要转义、还原的字符串进行一定范围的限制，通过使用 addcslashes()函数和 stripcslashes()函数实现对指定范围内的字符串进行自动转义、还原。下面分别对这两个函数进行详细介绍。

☑　addcslashes()函数

实现转义字符串中的字符，即在指定的字符 charlist 前加上反斜线。

语法格式如下：

```
string addcslashes (string str, string charlist)
```

参数 str 为将要被操作的字符串，参数 charlist 指定在字符串中的哪些字符前加上反斜线“\”，如果参数 charlist 中包含\n、\r 等字符，将以 C 语言风格转换，而其他非字母数字且 ASCII 码低于 32 以及高于 126 的字符均转换成八进制表示。



在定义参数 charlist 的范围时，需要明确在开始和结束的范围内的字符。

☑　stripcslashes()函数

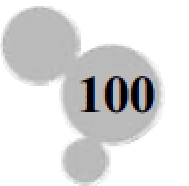
stripcslashes()函数用来将应用 addcslashes()函数转义的字符串 str 还原。

语法格式如下：

```
string stripcslashes (string str)
```

【例 5.6】 使用 addcslashes()函数对字符串“编程体验网”进行转义，使用 stripcslashes()函数对转义的字符串进行还原，实例代码如下：（**实例位置：光盘\TM\sl\5\6**）

```
<?php
$a="编程体验网";                                //对指定范围内的字符进行转义
echo $a;                                        //输出指定的字符串
```



```
echo "<br>";                                    //执行换行
$b=addcslashes($a,"编程体验网");                //转义指定的字符串
echo $b;                                        //输出转义后的字符串
echo "<br>";                                    //执行换行
$c=stripcslashes($b);                           //对转义的字符串进行还原
echo $c;                                        //输出还原后的转义字符串
?>
```

结果为：

```
编程体验网
\261\340\263\314\314\345\321\351\315\370
编程体验网
```



技巧

在缓存文件中，一般对缓存数据的值采用 addcslashes()函数进行指定范围的转义。

5.4.3　获取字符串的长度

获取字符串的长度使用的是 strlen()函数，下面重点讲解 strlen()函数的语法及其应用。

strlen()函数主要用于获取指定字符串 str 的长度。

语法格式如下：

```
int strlen(string str)
```

【例 5.7】 使用 strlen()函数来获取指定字符串的长度，实例代码如下：（**实例位置：光盘\TM\sl\5\7**）

```
<?php
    echo strlen("编程体验网:www.bcty365.com");              //输出指定字符串的长度
?>
```

结果为：26



说明

汉字占两个字符，数字、英文、小数点、下划线和空格占一个字符。

strlen()函数在获取字符串长度的同时，也可以用来检测字符串的长度。

【例 5.8】 使用 strlen()函数对提交的用户密码的长度进行检测，如果其长度小于 6，则弹出提示信息。（**实例位置：光盘\TM\sl\5\8**）

具体开发步骤如下：

（1）利用开发工具（如 Dreamweaver）新建一个 PHP 动态页，并将其保存为 index.php。

（2）添加一个表单，将表单的 action 属性设置为 index_ok.php。



（3）应用 HTML 标记设计页面，添加一个“用户名”文本框，命名为 user；添加一个“密码”文本框，命名为 pwd；添加一个图像域，指定源文件位置为 images/btn_dl.jpg。

（4）新建一个 PHP 动态页，保存为 index_ok.php，其代码如下：

```
<?php
if(strlen($_POST['pwd'])<6){                              //检测用户密码的长度是否小于 6
     echo "<script>alert('用户密码的长度不得少于 6 位!请重新输入'); history.back();</script>";
}
else{
     echo "用户信息输入合法！";                          //用户密码超过 6 位，则弹出该提示信息
}
?>
```

在上面的代码中，通过 POST 方法（关于 POST 方法将在后面的章节中进行详细讲解）接收用户输入的用户密码字符串的值。通过 strlen()函数来获取用户密码的长度，并使用 if 条件语句对用户密码长度进行判断，如果用户输入的密码没有达到这个长度，就会弹出提示信息。

（5）在 IE 浏览器中输入地址，按 Enter 键，运行结果如图 5.2 所示。



图 5.2　使用 strlen()函数检测字符串的长度

5.4.4　截取字符串

在 PHP 中有一项非常重要的技术，就是截取指定字符串中指定长度的字符。PHP 对字符串截取可以采用 PHP 的预定义函数 substr()实现。本节重点介绍字符串的截取技术。

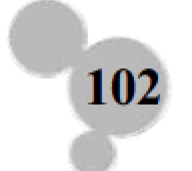
语法格式如下：

```
string substr ( string str, int start [, int length])
```

substr()函数的参数说明如表 5.2 所示。

表 5.2　substr()函数的参数说明

参　　数	说　　明
str	指定字符串对象
start	指定开始截取字符串的位置。如果参数 start 为负数，则从字符串的末尾开始截取
length	可选参数，指定截取字符的个数，如果 length 为负数，则表示取到倒数第 length 个字符





本函数中参数 start 的指定位置是从 0 开始计算的，即字符串中的第一个字符表示为 0。

【例 5.9】 使用 substr()函数截取字符串中指定长度的字符，实例代码如下：（**实例位置：光盘\TM\sl\5\9**）

```
<?php
echo substr("She is a well-read girl",0);            //从第 6 个字符开始截取
echo "<br>";                                          //执行换行
echo substr("She is a well-read girl",4,14);         //从第 4 个字符开始连续截取 14 个字符
echo "<br>";                                          //执行换行
echo substr("She is a well-read girl",-4,4);         //从倒数第 4 个开始截取 4 个字符
echo "<br>";                                          //执行换行
echo substr("She is a well-read girl",0,-4);         //从第一个字符开始截取，截取到倒数第 4 个字符
?>
```

结果为：

```
She is a well-read girl
is a well-read
girl
She is a well-read
```

在开发 Web 程序时，为了保持整个页面的合理布局，经常需要对一些超长文本进行部分显示。下面通过具体的实例讲解其实现方法。

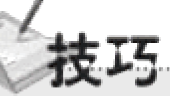
【例 5.10】 使用 substr()函数截取超长文本的部分字符串，剩余的部分用“…”代替，实例代码如下：（**实例位置：光盘\TM\sl\5\10**）

```
<?php
$text="祝全国程序开发人员在编程之路上一帆风顺二龙腾飞三羊开泰四季平安五福临门六六大顺七星高照八方来财九九同心十全十美百事可乐千事顺心万事吉祥 PHP 编程一级棒";
if(strlen($text)>30){                                 //如果文本的字符串长度大于 30
    echo substr($text,0,30)."...";                    //输出文本的前 30 个字节，然后输出省略号
}
else{                                                 //如果文本的字符串长度小于 30
    echo $text;                                       //直接输出文本
}
?>
```

结果为：祝全国程序开发人员在编程之路上……



从指定的字符串中按照指定的位置截取一定长度的字符。通过 substr()函数可以获取某个固定格式字符串中的一部分。


技巧

substr()函数在截取中文字符串时，如果截取的字符串中出现奇数，那么就会导致截取的中文字符串出现乱码，因为一个中文字符由两个字节组成。所以 substr()函数适用于对英文字符串的截取，如果想要对中文字符串进行截取，而且要避免出现乱码，最好的方法就是应用 substr()编写一个自定义函数。

5.4.5 比较字符串

在 PHP 中，对字符串之间进行比较的方法有很多种，第一种是使用 strcmp()函数按照字节进行比较，第二种是使用 strnatcmp()函数按照自然排序法进行比较，第三种是使用 strncmp()函数指定从源字符串的位置开始比较。下面分别对这 3 种方法进行详细讲解。

1. 按字节进行字符串的比较

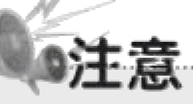
按字节进行字符串比较的方法有两种，分别是 strcmp()和 strcasecmp()函数，通过这两个函数即可实现对字符串进行按字节的比较。这两种函数的区别是 strcmp()函数区分字符的大小写，而 strcasecmp()函数不区分字符的大小写。由于这两个函数的实现方法基本相同，这里只介绍 strcmp()函数。

strcmp()函数用来对两个字符串进行比较。

语法格式如下：

```
int strcmp ( string str1, string str2)
```

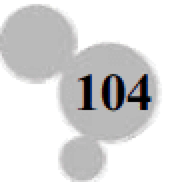
参数 str1 和参数 str2 指定要比较的两个字符串。如果相等则返回 0；如果参数 str1 大于参数 str2 则返回值大于 0；如果参数 str1 小于参数 str2 则返回值小于 0。


注意

该函数区分字母大小写。

【例 5.11】 使用 strcmp()函数和 strcasecmp()函数分别对两个字符串按字节进行比较，实例代码如下：**（实例位置：光盘\TM\sl\5\11）**

```
<?php
$str1="明日编程词典!";                    //定义字符串常量
$str2="明日编程词典!";                    //定义字符串常量
$str3="mrsoft";                           //定义字符串常量
$str4="MRSOFT";                           //定义字符串常量
echo strcmp($str1,$str2);                 //这两个字符串相等
echo strcmp($str3,$str4);                 //注意该函数区分大小写
echo strcasecmp($str3,$str4);             //该函数不区分字母大小写
?>
```



结果为：0　1　0


技巧

在 PHP 中，对字符串之间进行比较的应用也是非常广泛的。例如，使用 strcmp()函数比较在用户登录系统中输入的用户名和密码是否正确。如果在验证用户名和密码时不使用此函数，那么输入的用户名和密码无论是大写还是小写，只要正确即可登录。使用了 strcmp()函数之后就避免了这种情况，即使正确，也必须大小写匹配才可以登录，从而提高了网站的安全性。

2．按自然排序法进行字符串的比较

在 PHP 中，按照自然排序法进行字符串的比较是通过 strnatcmp()函数来实现的。自然排序法比较的是字符串中的数字部分，将字符串中的数字按照大小进行排序。

语法格式如下：

```
int strnatcmp ( string str1, string str2)
```

如果字符串相等则返回 0，如果参数 str1 大于参数 str2 则返回值大于 0；如果参数 str1 小于参数 str2 则返回值小于 0。本函数区分字母大小写。

注意

在自然运算法则中，2 比 10 小，而在计算机序列中，10 比 2 小，因为“10”中的第一个数字是“1”，它小于 2。

【例 5.12】 使用 strnatcmp()函数按自然排序法进行字符串的比较，实例代码如下：（**实例位置：光盘\TM\sl\5\12**）

```
<?php
$str1="str2.jpg";                    //定义字符串常量
$str2="str10.jpg";                   //定义字符串常量
$str3="mrsoft1";                     //定义字符串常量
$str4="MRSOFT2";                     //定义字符串常量
echo strcmp($str1,$str2);            //按字节进行比较，返回 1
echo strcmp($str3,$str4);            //按字节进行比较，返回 1
echo strnatcmp($str1,$str2);         //按自然排序法进行比较，返回-1
echo strnatcmp($str3,$str4);         //按自然排序法进行比较，返回 1
?>
```

结果为：1　1　-1　1

说明

按照自然运算法进行比较，还可以使用另一个与 strnatcmp()函数作用相同，但不区分大小写的 strnatcasecmp()函数。

3．指定从源字符串的位置开始比较

strncmp()函数用来比较字符串中的前 n 个字符。

语法格式如下：

```
int strncmp(string str1,string str2,int len)
```

如果字符串相等则返回 0，如果参数 str1 大于参数 str2 则返回值大于 0；如果参数 str1 小于参数 str2 则返回值小于 0。该函数区分字母大小写。

strncmp()函数的参数说明如表 5.3 所示。

表 5.3　strncmp()函数的参数说明

参　　数	说　　明
str1	指定参与比较的第一个字符串对象
str2	指定参与比较的第二个字符串对象
len	必要参数，指定每个字符串中参与比较字符的数量

【例 5.13】 使用 strncmp()函数比较字符串的前两个字符是否与源字符串相等，实例代码如下：**（实例位置：光盘\TM\sl\5\13）**

```
<?php
$str1="I like PHP !";                         //定义字符串常量
$str2="i am fine !";                          //定义字符串常量
echo strncmp($str1,$str2,2);                  //比较前两个字符
?>
```

结果为：-1

从上面的代码中可以看出，由于变量$str2 中的字符串的首字母为小写，与变量$str1 中的字符串不匹配，因此比较后的字符串返回值为-1。

5.4.6　检索字符串

在 PHP 中，提供了很多应用于字符串查找的函数，PHP 也可以像 Word 那样实现对字符串的查找功能。下面讲解常用的字符串检索技术。

1．使用 strstr()函数查找指定的关键字

获取一个指定字符串在另一个字符串中首次出现的位置到后者末尾的子字符串。如果执行成功，则返回剩余字符串（存在相匹配的字符）；如果没有找到相匹配的字符，则返回 false。

语法格式如下：

```
string strstr ( string haystack, string needle)
```

strstr()函数的参数说明如表 5.4 所示。

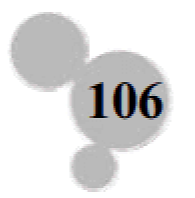


表 5.4　strstr()函数的参数说明

参　　数	说　　明
haystack	必要参数，指定从哪个字符串中进行搜索
needle	必要参数，指定搜索的对象。如果该参数是一个数值，那么将搜索与这个数值的 ASCII 值相匹配的字符



注意

本函数区分字母的大小写。

【例 5.14】 使用 strstr()函数获取上传图片的后缀，限制上传图片的格式。实例代码如下：（**实例位置：光盘\TM\sl\5\14**）

```
<form method="post" action="index.php" enctype="multipart/form-data">
	<input type="hidden" name="action" value="upload" />
	<input type="file" name="u_file"/>
	<input type="submit" value="上传" />
</form>
<?php
	if($_POST[action] == "upload"){                       //判断提交按钮是否为空
		$file_path = "./uploads\\";                     //定义图片在服务器中的存储位置
		$picture_name=$_FILES[u_file][name];            //获取上传图片的名称
		$picture_name=strstr($picture_name , ".");      //通过 strstr()函数截取上传图片的后缀
		if($picture_name!= ".jpg"){                     //根据后缀判断上传图片的格式是否符合要求
			echo "<script>alert('上传图片格式不正确,请重新上传'); window.location.href='index.php';</script>";
		}else if($_FILES[u_file][tmp_name]){
			move_uploaded_file($_FILES[u_file][tmp_name],$file_path.$_FILES[u_file][name]);  //执行图片上传
			echo "图片上传成功!";
		}
		else
			echo "上传图片失败";
		}
?>
```

运行结果如图 5.3 所示。



图 5.3　应用 strstr()函数检索上传图片的后缀



注意

strchr()函数与其正好相反，该函数是从字符串后序的位置开始检索子串（子字符串）的。

2. 使用 substr_count()函数检索子串出现的次数

获取指定字符在字符串中出现的次数。

语法格式如下：

```
int substr_count(string haystack,string needle)
```

参数 haystack 是指定的字符串，参数 needle 为指定的字符。

【例 5.15】 使用 substr_count()函数获取子串在字符串中出现的次数，实例代码如下：（**实例位置：光盘\TM\sl\5\15**）

```
<?php
$str="明日编程词典";                                  //输出查询的字符串
echo substr_count($str,"词");                         //输出查询的字符串
?>
```

结果为：1



注意

检索子串出现的次数一般常用于搜索引擎中，针对子串在字符串中出现的次数进行统计，便于用户第一时间掌握子串在字符串中出现的次数。

5.4.7 替换字符串

通过字符串的替换技术可以实现对指定字符串中的指定字符进行替换。字符串的替换技术可以通过以下两个函数实现：str_ireplace()函数和 substr_replace()函数。

1. str_ireplace()函数

使用新的子字符串（子串）替换原始字符串中被指定要替换的字符串。

语法格式如下：

```
mixed str_ireplace ( mixed search, mixed replace, mixed subject [, int &count])
```

将所有在参数 subject 中出现的参数 search 以参数 replace 取代，参数&count 表示取代字符串执行的次数。本函数区分大小写。

str_ireplace()函数的参数说明如表 5.5 所示。

表 5.5 str_ireplace()函数的参数说明

参　数	说　明
search	必要参数，指定需要查找的字符串
replace	必要参数，指定替换的值
subject	必要参数，指定查找的范围
count	可选参数，获取执行替换的数量



【例 5.16】 将文本中的指定字符串“某某”替换为“**”，并且输出替换后的结果，实例代码如下：（**实例位置：光盘\TM\sl\5\16**）

```
<?php
$str2="某某";                                    //定义字符串常量
$str1="**";                                      //定义字符串常量
$str="    某某公司是一家以计算机软件技术为核心的高科技企业，多年来始终致力于行业管理软件开发、数字化出版物制作、计算机网络系统综合应用以及行业电子商务网站开发等领域，涉及生产、管理、控制、仓贮、物流、营销、服务等行业";                   //定义字符串常量
echo str_ireplace($str2,$str1,$str);             //输出替换后的字符串
?>
```

结果为：

```
**公司是一家以计算机软件技术为核心的高科技企业，多年来始终致力于行业管理软件开发、数字化出版物制作、计算机网络系统综合应用以及行业电子商务网站开发等领域，涉及生产、管理、控制、仓贮、物流、营销、服务等行业
```

注意

该函数在执行替换的操作时不区分大小写，如果需要对大小写加以区分，可以使用 str_replace()函数。

字符串替换技术最常用的就是在搜索引擎的关键字处理中，可以使用字符串替换技术将搜索到的字符串中的关键字替换颜色，如查询关键字描红功能，使搜索到的结果更便于用户查看。

注意

查询关键字描红是指将查询关键字以特殊的颜色、字号或字体进行标识。这样可以使浏览者快速检索到所需的关键字，方便浏览者从搜索结果中查找所需内容。查询关键字描红适用于模糊查询。

下面通过具体的实例介绍如何实现查询关键字描红功能。

【例 5.17】 使用 str_ireplace()函数替换查询关键字，当显示所查询的相关信息时，将输出的关键字的字体替换为红色。实例代码如下：（**实例位置：光盘\TM\sl\5\17**）

```
<?php
$content="白领女子公寓，温馨街南行 200 米，交通便利，亲情化专人管理，您的理想选择！";
$str="女子公寓";                                                          //定义查询的字符串常量
echo str_ireplace($str,"<font color='#FF0000'>".$str."</font>",$content);  //替换字符串为红色字体
?>
```

运行结果如图 5.4 所示。

注意

查询关键字描红功能在搜索引擎中被广泛应用，希望读者通过本例的学习，能够举一反三，从而开发出更加灵活、便捷的程序。

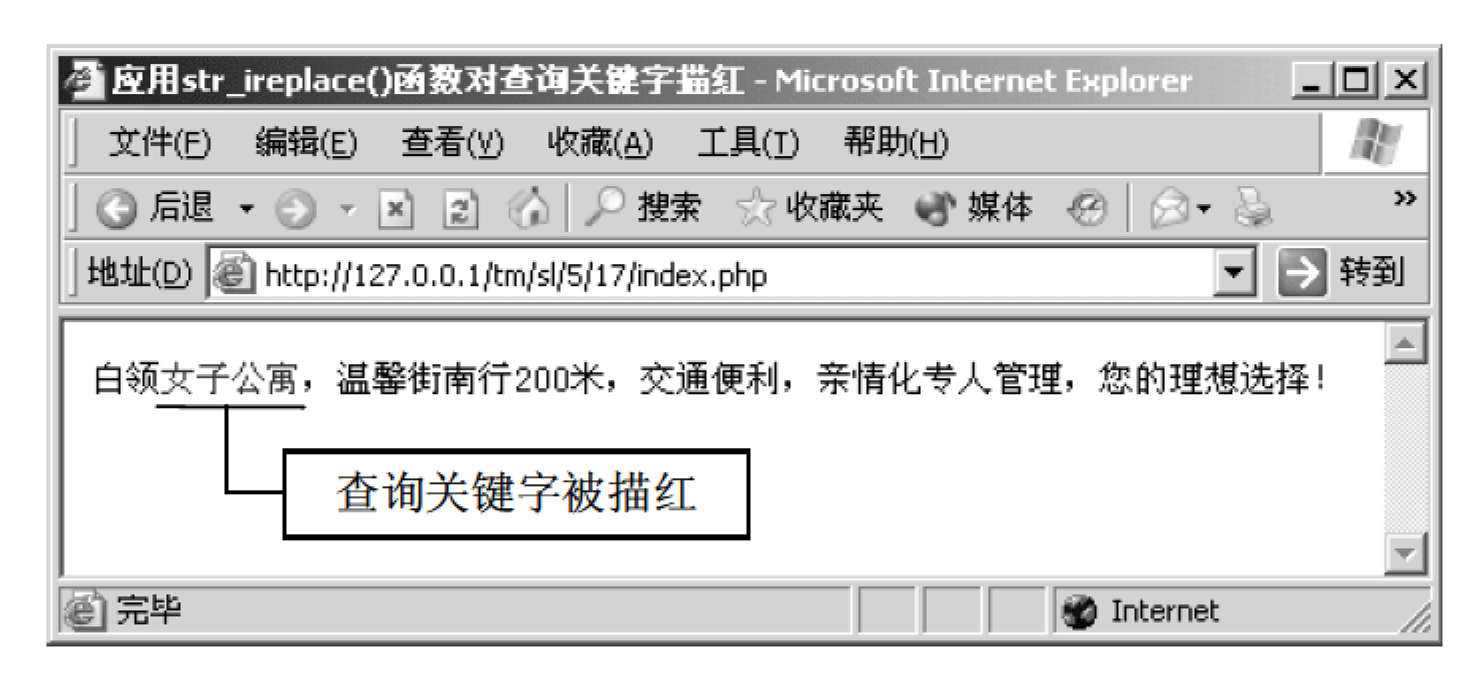


图 5.4　应用 str_ireplace()函数对查询关键描红

2．substr_replace()函数

对指定字符串中的部分字符串进行替换。

语法格式如下：

```
string substr_replace(string str,string repl,int start,[int length])
```

substr_replace()函数的参数说明如表 5.6 所示。

表 5.6　substr_replace()函数的参数说明

参　　数	说　　明
str	指定要操作的原始字符串
repl	指定替换后的新字符串
start	指定替换字符串开始的位置。正数表示起始位置从字符串开头开始；负数表示起始位置从字符串的结尾开始；0 表示起始位置从字符串中的第一个字符开始
length	可选参数，指定返回的字符串长度。默认值是整个字符串。正数表示起始位置从字符串开头开始；负数表示起始位置从字符串的结尾开始；0 表示插入而非替代

注意

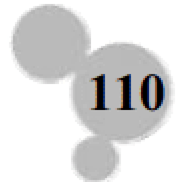
如果参数 start 设置为负数，而参数 length 数值小于或等于 start 数值，那么 length 的值自动为 0。

【例 5.18】　使用 substr_replace()函数对指定字符串进行替换，实例代码如下：（**实例位置：光盘\TM\sl\5\18**）

```
<?php
$str="用今日的辛勤工作，换明日的双倍回报！";                    //定义字符串常量
$replace="百倍";                                               //定义要替换的字符串
echo substr_replace($str,$replace,26,4);                       //替换字符串
?>
```

在上面的代码中，主要使用 substr_replace()函数实现将字符串“双倍”替换为字符串“百倍”。

结果为：用今日的辛勤工作，换明日的百倍回报！



5.4.8　格式化字符串

在 PHP 中，字符串的格式化方式有多种，按照格式化的类型可以分为字符串的格式化和数字的格式化，数字的格式化最为常用，本节将重点讲解数字格式化 number_format()函数。

number_format()函数用来将数字字符串格式化。

语法格式如下：

```
string number_format(float number,[int num_decimal_places],[string dec_seperator],string thousands_ seperator)
```

number_format()函数可以有一个、两个或是 4 个参数，但不能是 3 个参数。如果只有一个参数 number，number 格式化后会舍去小数点后的值，且每一千就会以逗号（，）来隔开；如果有两个参数，number 格式化后会到小数点第 num_decimal_places 位，且每一千就会以逗号来隔开；如果有 4 个参数，number 格式化后会到小数点第 num_decimal_places 位，dec_seperator 用来替代小数点（.），thousands_seperator 用来替代每一千隔开的逗号（，）。

【例 5.19】 使用 number_format()函数对指定的数字字符串进行格式化处理，实例代码如下：**（实例位置：光盘\TM\sl\5\19）**

```
<?php
$number = 1868.96;                              //定义数字字符串常量
echo number_format($number);                    //输出格式化后的数字字符串
echo "<br>";                                    //执行换行
echo number_format($number, 2);                 //输出格式化后的数字字符串
echo "<br>";                                    //执行换行
$number2 = 11886655.760055;                     //定义数字字符串常量
echo number_format($number2, 2, '.', '.');      //输出格式化后的数字字符串
?>
```

结果为：

```
1,869
1,868.96
11.886.655.76
```

5.4.9　分割字符串

字符串的分割是通过 explode()函数实现的。explode()函数按照指定的规则对一个字符串进行分割，返回值为数组。

语法格式如下：

```
array explode(string separator,string str,[int limit])
```

explode()函数的参数说明如表 5.7 所示。

表 5.7　explode()函数的参数说明

参　数	说　明
separator	必要参数，指定的分割符。如果 separator 为空字符串（""），explode()将返回 false。如果 separator 所包含的值在 str 中找不到，那么 explode()函数将返回包含 str 单个元素的数组
str	必要参数，指定将要被进行分割的字符串
limit	可选参数，如果设置了 limit 参数，则返回的数组包含最多 limit 个元素，而最后的元素将包含 string 的剩余部分；如果 limit 参数是负数，则返回除了最后的-limit 个元素外的所有元素

【例 5.20】 使用 explode()函数实现字符串分割，实例代码如下：（**实例位置：光盘\TM\sl\5\20**）

```
<?php
$str="PHP 编程词典@NET 编程词典@ASP 编程词典@JSP 编程词典";          //定义字符串常量
$str_arr=explode("@",$str);                                          //应用标识@分割字符串
print_r($str_arr);                                                   //输出字符串分割后的结果
?>
```

从上面的代码中可以看出，在分割字符$str 时，以“@”作为分割的标识符进行拆分，分割成 4 个数组元素，最后使用 print_r()输出函数输出数组中的元素。

运行结果如图 5.5 所示。



图 5.5　使用 explode()函数实现字符串分割

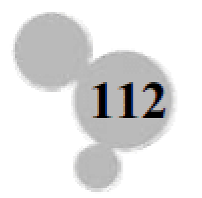
注意

在默认情况下，数组的第一个元素的索引为 0。关于数组的相关知识将在后续的章节中进行详细讲解。

输出数组元素除了使用 print_r()函数外，还可以使用 echo 语句进行输出，两者的区别是，print_r()函数输出的是一个数组列，而使用 echo 语句输出的是数组中的元素。将“print_r($str_arr);”使用如下代码替换即可输出数组中的元素。

```
echo $str_arr[0];                    //输出数组中的第一个元素
echo $str_arr[1];                    //输出数组中的第二个元素
echo $str_arr[2];                    //输出数组中的第 3 个元素
echo $str_arr[3];                    //输出数组中的第 4 个元素
```

结果为：PHP 编程词典 NET 编程词典 ASP 编程词典 JSP 编程词典





以上两种输出分割字符串的方法在运行结果的表现形式上会稍有不同。

5.4.10　合成字符串

implode()函数可以将数组的内容组合成一个新字符串。

语法格式如下：

```
string implode(string glue, array pieces)
```

参数 glue 是字符串类型，指定分隔符。参数 pieces 是数组类型，指定要被合并的数组。

【例 5.21】 应用 implode()函数将数组中的内容以@为分隔符进行连接，从而组合成一个新的字符串，实例代码如下：（**实例位置：光盘\TM\sl\5\21**）

```
<?php
	$str="PHP 编程词典@NET 编程词典@ASP 编程词典@JSP 编程词典";	//定义字符串常量
	$str_arr=explode("@",$str);	//应用标识@分割字符串
	$array=implode("@",$str_arr);	//将数组合成字符串
	echo $array;	//输出字符串
?>
```

结果为：PHP 编程词典@NET 编程词典@ASP 编程词典@JSP 编程词典



implode()函数和 explode()函数是两个相对的函数，一个用于合成，一个用于分隔。

5.5　小　　结

本章主要对常用的字符串操作技术进行了详细的讲解，其中去除字符串首尾空格、获取字符串的长度、连接和分割字符串、转义字符串、截取字符串和字符串的查找与替换等都是需要重点掌握的技术。同时，这些内容也是作为一个 PHP 程序员必须熟悉和掌握的知识。相信通过本章的学习，读者能够举一反三，对所学知识灵活运用，从而开发实用的 PHP 程序。

5.6　练习与实践

1．尝试开发一个页面，去除字符串“&& 明日编程词典 &&”首尾空格和特殊字符&&。（答

案位置：光盘\TM\sl\5\22）

2．尝试开发一个页面，验证用户输入的身份证号长度是否正确。**（答案位置：光盘\TM\sl\5\23）**

3．尝试开发一个页面，对检索到的用户输入的查询关键字进行加粗描红。**（答案位置：光盘\TM\sl\5\24）**

4．尝试开发一个页面，使用 explode()函数对全国各省会名称以逗号进行分割。**（答案位置：光盘\TM\sl\5\25）**

第 6 章

正则表达式

（视频讲解：27 分钟）

在新技术层出不穷的今天，让人难忘的、能称得上是伟大的却寥寥无几，其中一定会有正则表达式，然而，最容易被人忽略和让人遗忘的也是正则表达式。一方面，几乎所有的编程语言和文本编辑工具都支持正则表达式；另一方面，关于正则表达式的书籍、资料却少之又少。

通过阅读本章，您可以：

- 了解正则表达式的发展及相关概念
- 了解 PHP 中的 POSIX 函数
- 了解 PHP 中的 PERL 函数
- 掌握正则表达式的应用

6.1 什么是正则表达式

视频讲解：光盘\TM\lx\6\什么是正则表达式.exe

正则表达式是一种描述字符串结构的语法规则，是一个特定的格式化模式，可以匹配、替换、截取匹配的字串。对于用户来说，可能以前接触过 DOS，如果想匹配当前文件夹下所有的文本文件，可以输入“dir *.txt”命令，按 Enter 键后所有“.txt”文件将会被列出来。这里的“*.txt”即可理解为一个简单的正则表达式。

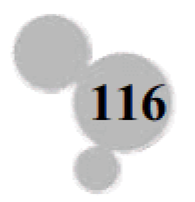
在学习正则表达式之前，先来了解一下正则表达式中的几个容易混淆的术语，这对于学习正则表达式有很大的帮助。

- ☑ grep：最初是 ED 编辑器中的一条命令，用来显示文件中特定的内容，后来成为一个独立的工具 grep。
- ☑ egrep：grep 虽然不断地更新升级，但仍然无法跟上技术的脚步。为此，贝尔实验室推出了 egrep，意为“扩展的 grep”，这大大增强了正则表达式的能力。
- ☑ POSIX（Portable Operating System Interface of Vnix）：可移植操作系统接口。在 grep 发展的同时，其他一些开发人员也按照自己的喜好开发出了具有独特风格的版本。但问题也随之而来，有的程序支持某个元字符，而有的程序则不支持。因此就有了 POSIX，POSIX 是一系列标准，确保了操作系统之间的可移植性。但 POSIX 和 SQL 一样，没有成为最终的标准而只能作为一个参考。
- ☑ Perl（Practical Extraction and Reporting Language）：实际抽取与汇报语言。1987 年，Larry Wall 发布了 Perl。在随后的 7 年时间里，Perl 经历了从 Perl1 到现在的 Perl5 的发展，最终 Perl 成为了 POSIX 之后的另一个标准。
- ☑ PCRE：Perl 的成功，让其他的开发人员在某种程度上要兼容 Perl，包括 C/C++、Java、Python 等都有自己的正则表达式。1997 年，Philip Hazel 开发了 PCRE 库，这是兼容 Perl 正则表达式的一套正则引擎，其他开发人员可以将 PCRE 整合到自己的语言中，为用户提供丰富的正则功能。许多软件都使用 PCRE，PHP 正是其中之一。

6.2 正则表达式语法规则

视频讲解：光盘\TM\lx\6\正则表达式语法规则.exe

一个完整的正则表达式由两部分构成，元字符和文本字符。元字符就是具有特殊含义的字符，如前面提到的“*”和“？”。文本字符就是普通的文本，如字母和数字等。PCRE 风格的正则表达式一般都放置在定界符“/”中间。如“^\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/”、“/^http:\/\/ (www\.)?.+.?$/”。为了便于读者理解，除了个别实例外，本节中的表达式不给出定界符“/”。更多的关于定界符的应用请参考 6.4 节。



6.2.1　行定位符（^和$）

行定位符就是用来描述字串的边界。“^”表示行的开始；“$”表示行的结尾。如：

```
^tm
```

该表达式表示要匹配字串 tm 的开始位置是行头，如 tm equal Tomorrow Moon 就可以匹配，而 Tomorrow Moon equal tm 则不匹配。但如果使用

```
tm$
```

则后者可以匹配而前者不能匹配。如果要匹配的字串可以出现在字符串的任意部分，那么可以直接写成

```
tm
```

这样两个字符串就都可以匹配了。

6.2.2　单词定界符（\b、\B）

继续上面的实例，使用 tm 可以匹配在字符串中出现的任何位置。那么类似 html、utmost 中的 tm 也会被查找出来。但现在需要匹配的是单词 tm，而不是单词的一部分。这时可以使用单词分界符\b，表示要查找的字串为一个完整的单词。如：

```
\btm\b
```

还有一个大写的\B，意思和\b 相反。它匹配的字串不能是一个完整的单词，而是其他单词或字串的一部分。如：

```
\Btm\B
```



说明

关于反斜线的用法，请参考 6.2.10 节。

6.2.3　字符类（[ ]）

正则表达式是区分大小写的，如果要忽略大小写可使用方括号表达式“[]”。只要匹配的字符出现在方括号内，即可表示匹配成功。但要注意：一个方括号只能匹配一个字符。例如，要匹配的字串 tm 不区分大小写，那么该表达式应该写作如下格式：

```
[Tt][Mm]
```

这样，即可匹配字串 tm 的所有写法。POSIX 和 PCRE 都使用了一些预定义字符类。但表示方法略有不同。POSIX 风格的预定义字符类如表 6.1 所示。

表 6.1　POSIX 预定义字符类

预定义字符类	说　明
[:digit:]	十进制数字集合。等同于[0-9]
[[:alnum:]]	字母和数字的集合。等同于[a-zA-Z0-9]
[[:alpha:]]	字母集合。等同于[a-zA-Z]
[[:blank:]]	空格和制表符
[[:xdigit:]]	十六进制数字
[[:punct:]]	特殊字符集合。包括键盘上的所有特殊字符，如!@#$?等
[[:print:]]	所有的可打印字符（包括空白字符）
[[:space:]]	空白字符（空格、换行符、换页符、回车符、水平制表符）
[[:graph:]]	所有的可打印字符（不包括空白字符）
[[:upper:]]	所有大写字母，[A-Z]
[[:lower:]]	所有小写字母，[a-z]
[[:cntrl:]]	控制字符

而 PCRE 的预定义字符类则使用反斜线来表示，请参考 6.2.10 节。

6.2.4　选择字符（|）

还有一种方法可以实现上面的匹配模式，就是使用选择字符（|）。该字符可以理解为“或”，如上例也可以写成

```
(T|t)(M|m)
```

该表达式的意思是以字母 T 或 t 开头，后面接一个字母 M 或 m。

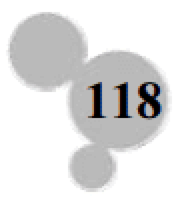

说明

使用“[]”和使用“|”的区别在于“[]”只能匹配单个字符，而“|”可以匹配任意长度的字串。如果不怕麻烦，上例还可以写为

```
TM|tm|Tm|tM
```

6.2.5　连字符（-）

变量的命名规则是只能以字母和下划线开头。但这样一来，如果要使用正则表达式来匹配变量名



的第一个字母，要写为

```
[a,b,c,d…A,B,C,D…]
```

这无疑是非常麻烦的，正则表达式提供了连字符“-”来解决这个问题。连字符可以表示字符的范围。如上例可以写成

```
[a-zA-Z]
```

6.2.6　排除字符（[^]）

上面的例子是匹配符合命名规则的变量。现在反过来，匹配不符合命名规则的变量，正则表达式提供了“^”字符。这个元字符在 6.2.1 节中出现过，表示行的开始。而这里将会放到方括号中，表示排除的意思。例如：

```
[^a-zA-Z]
```

该表达式匹配的就是不以字母和下划线开头的变量名。

6.2.7　限定符（? * + {n,m}）

经常使用 google 的用户可能会发现，在搜索结果页的下方，google 中间字母 o 的个数会随着搜索页的改变而改变。那么要匹配该字串的正则表达式该如何实现呢？

对于这类重复出现字母或字串，可以使用限定符来实现匹配。限定符主要有 6 种，如表 6.2 所示。

表 6.2　限定符的说明和举例

限　定　符	说　　明	举　　例
?	匹配前面的字符零次或一次	colou?r，该表达式可以匹配 colour 和 color
+	匹配前面的字符一次或多次	go+gle，该表达式可以匹配的范围从 gogle 到 goo…gle
*	匹配前面的字符零次或多次	go*gle，该表达式可以匹配的范围从 ggle 到 goo…gle
{n}	匹配前面的字符 n 次	go{2}gle，该表达式只匹配 google
{n,}	匹配前面的字符最少 n 次	go{2,}gle，该表达式可以匹配的范围从 google 到 goo…gle
{n,m}	匹配前面的字符最少 n 次，最多 m 次	employe{0,2}，该表达式可以匹配 employ、employe 和 employee 3 种情况

可以发现，在表 6.1 中实际已经对字符串进行了匹配，只是还不完善。通过观察发现，当 google 搜索结果只有一页时，不显示 google 标志，只有大于等于 2 时，才显示 google。说明字母 o 最少为两个，最多为 20 个，那么正则表达式为：

```
go{2,20}gle
```

6.2.8　点号字符（.）

如遇到这样的试题：写出 5～10 个以 s 开头、t 结尾的单词，这是有很大难度的。如果考题并不告知第一个字母，而是中间任意一个。无疑难度会更大。

在正则表达式中可以通过点字符（.）来实现这样的匹配。点字符（.）可以匹配出换行符外的任意一个字符。注意：是除了换行符外的、任意的一个字符。如匹配以 s 开头、t 结尾、中间包含一个字母的单词。格式如下：

```
^s.t$
```

匹配的单词包括：sat、set、sit 等。再举一个实例，匹配一个单词，它的第一个字母为 r，第 3 个字母为 s，最后一个字母为 t。能匹配该单词的正则表达式为：

```
^r.s.*t$
```

6.2.9　转义字符（\）

正则表达式中的转移字符（\）和 PHP 中的大同小异，都是将特殊字符（如“.”、“?”、“\”等）变为普通的字符。举一个 IP 地址的实例，用正则表达式匹配诸如 127.0.0.1 这样格式的 IP 地址。如果直接使用点字符，格式为：

```
[0-9]{1,3}(.[0-9]{1,3}){3}
```

这显然不对，因为“.”可以匹配一个任意字符。这时，不仅是 127.0.0.1 这样的 IP，连 127101011 这样的字串也会被匹配出来。所以在使用“.”时，需要使用转义字符（\）。修改后上面的正则表达式格式为：

```
[0-9]{1,3}(\.[0-9]{1,3}){3}
```


说明

括号在正则表达式中也算是一个元字符，关于括号的作用请参考 6.2.11 节。

6.2.10　反斜线（\）

除了可以做转义字符外，反斜线还有其他一些功能。

☑　反斜线可以将一些不可打印的字符显示出来，如表 6.3 所示。

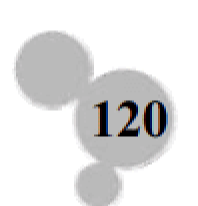


表 6.3　反斜线显示的不可打印字符

字　符	说　明
\a	警报，即 ASCII 中的<BEL>字符（0x07）
\b	退格，即 ASCII 中的<BS>字符（0x08）。注意，在 PHP 中只有在中括号（[ ]）里使用才表示退格
\e	Escape，即 ASCII 中的<ESC>字符（0x1B）
\f	换页符，即 ASCII 中的<FF>字符（0x0C）
\n	换行符，即 ASCII 中的<LF>字符（0x0A）
\r	回车符，即 ASCII 中的<CR>字符（0x0D）
\t	水平制表符，即 ASCII 中的<HT>字符（0x09）
\xhh	十六进制代码
\ddd	八进制代码
\cx	即 control-x 的缩写，匹配由 x 指明的控制字符，其中 x 是任意字符

☑ 还可以指定预定义字符集，如表 6.4 所示。

表 6.4　反斜线指定的预定义字符集

预定义字符集	说　明
\d	任意一个十进制数字，相当于[0-9]
\D	任意一个非十进制数字
\s	任意一个空白字符（空格、换行符、换页符、回车符、水平制表符），相当于[\f\n\r\t]
\S	任意一个非空白字符
\w	任意一个单词字符，相当于[a-zA-Z0-9_]
\W	任意一个非单词字符

☑ 反斜线还有一种功能，就是定义断言，其中已经了解过了\b、\B，其他如表 6.5 所示。

表 6.5　反斜线定义断言的限定符

限　定　符	说　明
\b	单词分界符，用来匹配字符串中的某些位置，\b 是以统一的分界符来匹配
\B	非单词分界符序列
\A	总是能够匹配待搜索文本的起始位置
\Z	表示在未指定任何模式下匹配的字符，通常是字符串的末尾位置，或者是在字符串末尾的换行符之前的位置
\z	只匹配字符串的末尾，而不考虑任何换行符
\G	当前匹配的起始位置

6.2.11　括号字符（()）

通过 6.2.4 节的实例，相信读者已经对小括号的作用有了一定的了解。这里，再通过几个实例来巩

固一下对小括号字符的印象。

小括号字符的第一个作用就是可以改变限定符的作用范围，如“|”、“*”、“^”等。来看下面的一个表达式。

```
(thir|four)th
```

这个表达式的意思是匹配单词 thirth 或 fourth，如果不使用小括号，那么就变成了匹配单词 thir 和 fourth 了。

小括号的第二个作用是分组，也就是子表达式。如(\.[0-9]{1,3}){3}，就是对分组(\.[0-9]{1,3})进行重复操作。后面要学到的反向引用和分组有着直接的关系。

6.2.12　反向引用

反向引用，就是依靠子表达式的“记忆”功能来匹配连续出现的字串或字母。如匹配连续两个 it，首先将单词 it 作为分组，然后在后面加上“\1”即可。格式为：

```
(it)\1
```

这就是反向引用最简单的格式。如果要匹配的字串不固定，那么就将括号内的字串写成一个正则表达式。如果使用了多个分组，那么可以用“\1”、“\2”来表示每个分组（顺序是从左到右）。如：

```
([a-z])([A-Z])\1\2
```

除了可以使用数字来表示分组外，还可以自己来指定分组名称。语法格式如下：

```
(?P<subname>…)
```

如果想要反向引用该分组，使用如下语法：

```
(?P=subname)
```

下面来重写一下表达式([a-z])([A-Z])\1\2。为这两个分组分别命名，并反向引用它们。正则表达式如下：

```
(?P<fir>[a-z])(?P<sec>[A-Z])(?P=fir)(?P=sec)
```

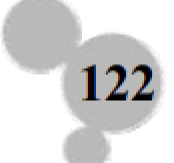
反向引用还可以参考 6.4.4 节

6.2.13　模式修饰符

模式修饰符的作用是设定模式。也就是规定正则表达式应该如何解释和应用。不同的语言都有自己的模式设置，PHP 中的主要模式如表 6.6 所示。

表 6.6　模式修饰符

修　饰　符	表达式写法	说　　明
i	(?i)…(?-i)、(?i:…)	忽略大小写模式
M	(?m)…(?-m)、(?m:…)	多文本模式。即字串内部有多个换行符时，影响“^”和“$”的匹配



续表

修　饰　符	表达式写法	说　　明
s	(?s)…(?-s)、(?s:…)	单文本模式。在此模式下，元字符点号（.）可以匹配换行符。其他模式则不能匹配换行符
X	(?x)…(?-x)、(?x:…)	忽略空白字符

模式修饰符既可以写在正则表达式的外面，也可以写在表达式内。如忽略大小写模式，可以写为/tm/i、(?i)tm(?-i)和(?i:tm)3 种格式。

6.3　POSIX 扩展正则表达式函数

视频讲解：光盘\TM\lx\6\POSIX 正则表达式.exe

PHP 中实现 POSIX 正则表达式的函数有 7 个。首先来了解一下主要函数的语法。

6.3.1　ereg()函数和 eregi()函数

函数语法：

```
bool ereg/eregi ( string pattern, string string [, array regs] )
```

函数功能：在字符串 string 中匹配表达式 pattern，如果匹配成功返回 true，否则返回 false。如果有第 3 个参数 regs，则将成功匹配的字串按子串（子表达式）划分，并存储到 regs 数组中。ereg 区分大小写，而 eregi 不区分大小写。

【例 6.1】 使用 ereg()函数验证变量是否合法。实例代码如下：**（实例位置：光盘\TM\sl\6\1）**

```
<?php
$ereg = '^[$][[:alpha:]_][[:alnum:]]*';                //要匹配的字串
ereg($ereg,'$_name',$register);                         //使用 ereg()函数匹配变量名
var_dump($register);                                    //显示数组结构
?>
```

结果为：array(1) { [0]=> string(6) "$_name" }

6.3.2　ereg_replace()函数和 eregi_replace()函数

函数语法：

```
string ereg_replace/eregi_replace ( string pattern, string replacement, string string )
```

函数功能：在字符串 string 中匹配表达式 pattern。如果匹配成功，则使用 replacement 来替换匹配字串，并返回替换后的 string。eregi_replace()不区分大小写。

【例 6.2】 将字符串中所有非大写的 tm 都换成大写 TM。实例代码如下：（**实例位置：光盘\TM\sl\6\2**）

```
<?php
	$ereg = 'tm';							//要匹配的字串
	$str = 'hello,tm,Tm,tM.';					//要查找的文本
	$rep_str = eregi_replace($ereg,'TM',$str);		//使用 eregi_replace()函数进行替换
	echo $rep_str;							//输出替换后的文本
?>
```

结果为：hello,TM,TM,TM.

6.3.3　split()函数和 spliti()函数

函数语法：

```
array split/spliti ( string pattern, string string [, int limit] )
```

函数功能：使用表达式 pattern 来分割字符串 string。如果有参数 limit，那么数组最多有 limit 个元素，剩余部分都写到最后一个数组元素中。如果函数错误，则返回 false。split()函数区分大小写，spliti()函数不区分大小写。

【例 6.3】 本例使用字串 is 来分割字符串$str，了解 split()函数对空格的处理方法。实例代码如下：（**实例位置：光盘\TM\sl\6\3**）

```
<?php
	$ereg = 'is';							//分割字符串使用的表达式
	$str = 'This is a register book.';			//要被分割的字符串
	$arr_str = spliti($ereg,$str);				//使用 spliti()分割字符串
	var_dump($arr_str);						//显示分割后的数组结构
?>
```

结果为：array(4){[0]=>string(2)"Th"[1]=>string(1)""[2]=>string(6)"a reg"[3]=>string(9)"ter book."}

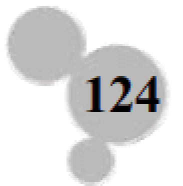
6.4　PCRE 兼容正则表达式函数

视频讲解：光盘\TM\lx\6\PCRE 正则表达式.exe

实现 PCRE 风格的正则表达式的函数也有 7 个。但无论从执行效率还是从语法支持上，PCRE 函数都要略优于 POSIX 函数。下面就来了解一下这 7 个 PCRE 函数。

6.4.1　preg_grep()函数

函数语法：



```
array preg_grep ( string pattern, array input )
```

函数功能：使用数组 input 中的元素一一匹配表达式 pattern，最后返回由所有相匹配的元素所组成的数组。

【例 6.4】 在数组$arr 中匹配具有正确格式的电话号（010-1234****等），并保存到另一个数组中。实例代码如下：**（实例位置：光盘\TM\sl\6\4）**

```
<?php
    $preg = '/\d{3,4}-?\d{7,8}/';                                    //国内电话格式表达式
    $arr = array('043212345678','0431-7654321','12345678');          //包含元素的数组
    $preg_arr = preg_grep($preg,$arr);                               //使用 preg_grep()查找匹配元素
    var_dump($preg_arr);                                             //查看新数组结构
?>
```

结果为：array(2) { [0]=> string(12) "043212345678" [1]=> string(12) "0431-7654321" }

6.4.2　preg_match()函数和 preg_match_all()函数

函数语法：

```
int preg_match/preg_match_all ( string pattern, string subject [, array matches] )
```

函数功能：在字符串 subject 中匹配表达式 pattern。函数返回匹配的次数。如果有数组 matches，那么每次匹配的结果都将被存储到数组 matches 中。

函数 preg_match()的返回值是 0 或 1。因为该函数在匹配成功后就停止继续查找了。而 preg_match_all()函数则会一直匹配到最后才会停止。参数 array matches 对于 preg_match_all()函数是必须有的，而对前者则可以省略。

【例 6.5】 使用 preg_match()函数和 preg_match_all()函数来匹配字串$str，并返回各自的匹配次数。实例代码如下：**（实例位置：光盘\TM\sl\6\5）**

```
<?php
    $str = 'This is an example!';
    $preg = '/\b\w{2}\b/';
    $num1 = preg_match($preg,$str,$str1);
    echo $num1.'<br>';
    var_dump($str1);
    $num2 = preg_match_all($preg,$str,$str2);
    echo '<p>'.$num2.'<br>';
    var_dump($str2);
?>
```

运行结果如图 6.1 所示。

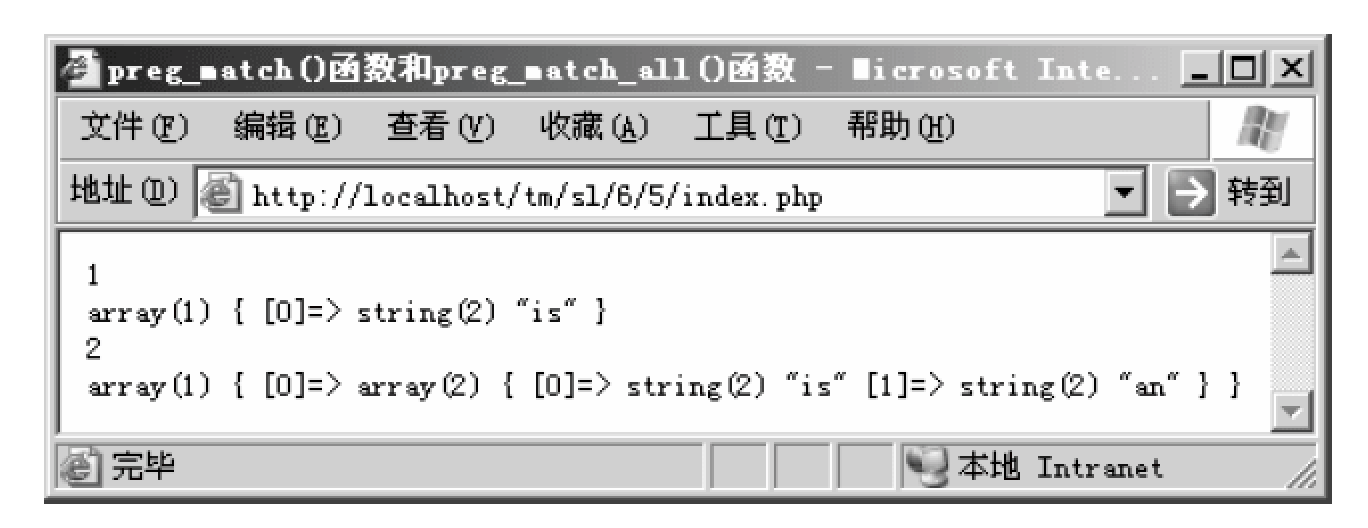


图 6.1 preg_match()函数和 preg_match_all()函数

6.4.3 preg_quote()函数

函数语法：

```
string preg_quote ( string str [, string delimiter] )
```

函数功能：该函数将字符串 str 中的所有特殊字符进行自动转义。如果有参数 delimiter，那么该参数所包含的字串也将被转义。函数返回转义后的字串。

【例 6.6】 输出常用的特殊字符，并且将字母 b 也当作特殊字符输出。实例代码如下：**（实例位置：光盘\TM\sl\6\6）**

```
<?php
	$str = '!、$、^、*、+、.、[、]、\\、/、b、<、>';
	$str2 = 'b';
	$match_one = preg_quote($str,$str2);
	echo $match_one;
?>
```

结果为：\!、\$、\^、*、\+、\.、\[、\]、\\、/、\b、\<、\>

注意

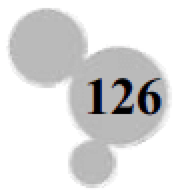
这里的特殊字符是指正则表达式中具有一定意义的元字符。其他如“@”、“#”等则不会被当作特殊字符处理。

6.4.4 preg_replace()函数

函数语法：

```
mixed preg_replace ( mixed pattern, mixed replacement, mixed subject [, int limit] )
```

函数功能：该函数在字符串 subject 中匹配表达式 pattern，并将匹配项替换成字串 replacement。如果有参数 limit，则替换 limit 次。



说明

如果参数中调用的是数组，有可能在调用过程中并不是按照数组的 key 值进行替换，所以在调用之前需要将数组重新排列 ksort()。

【例 6.7】 本例实现一个常见的 UBB 代码转换功能，将输入的“[b]…[/b]”、“[i]…[/i]”等类似的格式转换为 html 能识别的标签。实例代码如下：**（实例位置：光盘\TM\sl\6\7）**

```
<?php
	$string = '[b]粗体字[/b]';
	$b_rst = preg_replace('/\[b\](.*)\[\/b\]/i','<b>$1</b>',$string);
	echo $b_rst;
?>
```

结果为：粗体字



说明

preg_replace()函数中的字串“$1”是在正则表达式外调用分组，按照$1、$2 排列，依次表示从左到右的分组顺序，也就是括号顺序。$0 表示的是整个正则表达式的匹配值。关于反向引用的其他用法，请参考 6.2.12 节。

6.4.5　preg_replace_callback()函数

函数语法：

```
mixed preg_replace_callback ( mixed pattern, callback callback, mixed subject [, int limit] )
```

preg_replace_callback()函数与 preg_replace()函数的功能相同，都用于查找和替换字串。不同的是 preg_replace_callback()函数使用一个回调函数（callback）来代替 replacement 参数。

注意

在 preg_replace_callback()函数的回调函数中，字符串使用''，这样可以保证字符串中的特殊符号不被转义。

【例 6.8】 本例使用回调函数来实现 UBB 功能。实例代码如下：**（实例位置：光盘\TM\sl\6\8）**

```
<?php
function c_back($str){
	$str = "<font color=$str[1]>$str[2]</font>";
	return $str;
}
$string = '[color=blue]字体颜色[/color]';
```

```
echo preg_replace_callback('/\[color=(.*)\](.*)\[\/color\]/U',"c_back",$string);
?>
```

结果为：字体颜色



注意

本实例运行结果“字体颜色”为蓝色字体，书中看不出效果，请运行本书光盘附带的实例。

6.4.6 preg_split()函数

函数语法：

```
array preg_split ( string pattern, string subject [, int limit ] )
```

函数功能：使用表达式 pattern 来分割字符串 subject。如果有参数 limit，那么数组最多有 limit 个元素。该函数与 ereg_split()函数的使用方法相同，这里不再举例。

6.5 应用正则表达式对用户注册信息进行验证

【例 6.9】 通过正则表达式对用户注册信息的合理性进行判断，对用户输入的邮编、电话号码、邮箱地址和网址的格式进行判断。本实例中应用正则表达式和 JavaScript 脚本，判断用户输入信息的格式是否正确。实例代码如下：**（实例位置：光盘\TM\sl\6\9）**

首先，在 index.php 页面中通过 Script 脚本调用 js 脚本文件 check.js，创建 form 表单，实现会员注册信息的提交，并应用 onSubmit 事件调用 chkreg()方法对表单元素中的数据进行验证，将数据提交到 index_ok.php 文件中。index.php 的关键代码如下：

```
<script src="js/check.js"></script>
<form name="reg_check" method="post" action="index_ok.php" onSubmit="return chkreg(reg_check,'all')">
<table width="550" height="270" border="0" align="center" cellpadding="0" cellspacing="0">
    <tr>
        <td height="30"><div align="right">邮政编码：</div></td>
        <td height="30" colspan="2" align="left"> 
            <input type="text" name="postalcode" size="20"   onBlur="chkreg(reg_check,2)">
            <div id="check_postalcode" style="color:#F1B000"></div>
        </td>
    </tr>
    <tr>
        <td height="30"><div align="right">E-mail：</div></td>
        <td height="30" colspan="2" align="left"> 
        <input type="text" name="email" size="20" onBlur="chkreg(reg_check,4)">
            <font color="#999999">请务必正确填写您的邮箱</font>
```

```
                <div id="check_email" style="color:#F1B000"></div>
            </td>
    </tr>
    <tr>
        <td height="30" align="right">固定电话：</td>
        <td height="30" colspan="2" align="left"> 
            <input type="text" name="gtel" size="20"   onBlur="chkreg(reg_check,6)">
            <font color="#999999"><div id="check_gtel" style="color:#F1B000"></div></font></td>
    </tr>
    <tr>
        <td height="30"><div align="right">移动电话：</div></td>
        <td height="30" colspan="2" align="left"> 
             <input type="text" name="mtel" size="20"   onBlur="chkreg(reg_check,5)">
             <div id="check_mtel" style="color:#F1B000"></div></td>
    </tr>
    <tr>
        <td width="100" height="30"><input type="image"   src="images/bg_09.jpg"></td>
        <td  width="340"><img  src="images/bg_11.jpg"  width="56"  height="30"  onClick="reg_check.reset()"
style="cursor:hand"/></td>
    </tr>
</table>
</form>
```

在 check.js 脚本文件中，创建自定义方法，应用正则表达式对会员注册的电话号码和邮箱进行验证。其关键代码如下：

```
function checkregtel(regtel){
    var str=regtel;
    var Expression=/^13(\d{9})$|^18(\d{9})$|^15(\d{9})$/;                  //验证手机号码
    var objExp=new RegExp(Expression);
    if(objExp.test(str)==true){
        return true;
    }else{
        return false;
    }
}
function checkregtels(regtels){
    var str=regtels;
    var Expression=/^(\d{3}-)(\d{8})$|^(\d{4}-)(\d{7})$|^(\d{4}-)(\d{8})$/;   //验证座机号码
    var objExp=new RegExp(Expression);
    if(objExp.test(str)==true){
        return true;
    }else{
        return false;
    }
}
function checkregemail(emails){
    var str=emails;
    var Expression=/\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/;            //验证邮箱地址
    var objExp=new RegExp(Expression);
```

```
        if(objExp.test(str)==true){
            return true;
        }else{
            return false;
        }
}
```

运行结果如图 6.2 所示。



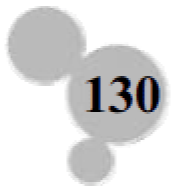
图 6.2　应用正则表达式对用户注册信息进行验证

注意

在本实例中通过正则表达式对表单提交的数据进行验证，在 JavaScript 脚本中，应用 onBlur 事件调用对应的方法对表单提交的数据直接进行验证，并通过 div 标签返回结果。

6.6　小　　结

本章介绍了什么是正则表达式和正则表达式的发展情况。接着介绍了正则表达式的两种主要风格和相关的 PHP 函数。最后应用正则表达式简单实现了一个用户注册信息验证实例。相信通过本章的学习，读者可以初步掌握正则表达式，并在以后的学习和工作中逐步提高自己的水平。



6.7　练习与实践

1．应用正则表达式实现 UBB 使用帮助。**（答案位置：光盘\TM\sl\6\10）**

2．使用正则表达式匹配 Email 地址标签。**（答案位置：光盘\TM\sl\6\11）**

3．使用正则表达式匹配 html 标签。**（答案位置：光盘\TM\sl\6\12）**

第 7 章

PHP 数组

（视频讲解：1 小时 9 分钟）

数组是对大量数据进行有效组织和管理的手段之一，通过数组的强大功能，可以对大量性质相同的数据进行存储、排序、插入及删除等操作，从而可以有效地提高程序开发效率及改善程序的编写方式。PHP 作为市面上最为流行的 Web 开发语言之一，凭借其代码开源、升级速度快等特点，对数组的操作能力更为强大，尤其是 PHP 为程序开发人员提供了大量方便、易懂的数组操作函数，更使 PHP 深受广大 Web 开发人员的青睐。

通过阅读本章，您可以：

- 了解数组的概念
- 掌握声明一维数组和二维数组的方法
- 掌握如何输出数组
- 掌握遍历数组的方法
- 掌握进行字符串与数组之间的转换方法
- 熟悉如何统计数组元素个数
- 熟悉查询数组中指定元素
- 掌握如何获取数组中最后一个元素
- 掌握如何向数组中添加元素
- 掌握如何删除数组中重复元素
- 熟悉数组函数在多文件上传中的应用

7.1 什么是数组

视频讲解：光盘\TM\lx\7\了解数组.exe

数组就是一组数据的集合，把一系列数据组织起来，形成一个可操作的整体。PHP 中的数组较为复杂，但比其他许多高级语言中的数组更灵活。数组 array 是一组有序的变量，其中每个变量被称为一个元素。每个元素由一个特殊的标识符来区分，这个标识符称为键（也称为下标）。数组中的每个实体都包含两项：键和值。可以通过键值来获取相应数组元素，这些键可以是数值键或关联键。如果说变量是存储单个值的容器，那么数组就是存储多个值的容器。数组结构如图 7.1 所示。

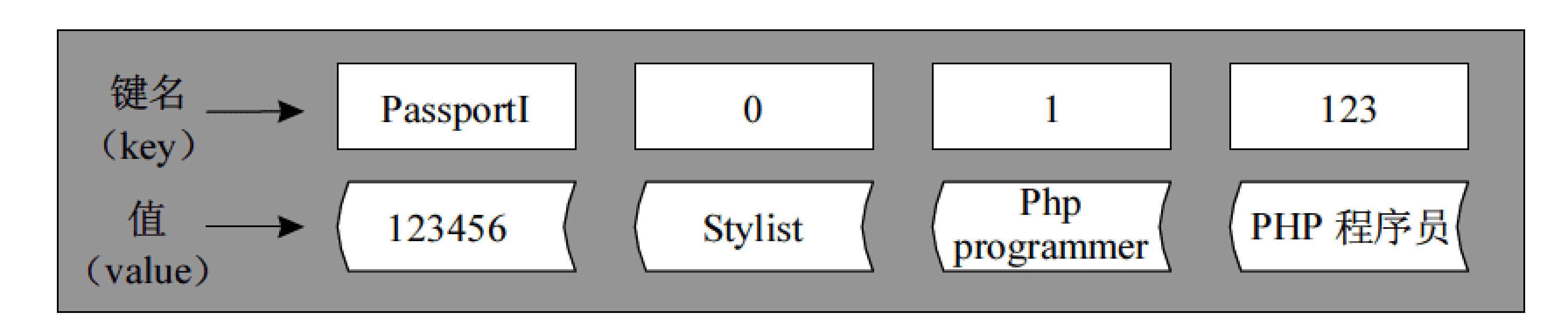


图 7.1　PHP 的数组结构

例如，一个足球队通常会有几十个人，但认识他们时首先会把他们看作是某队的成员，然后再利用他们的号码来区分每一名队员，这时，球队就是一个数组，而号码就是数组的下标，当指明是几号队员时就找到了这名队员。

7.2 声 明 数 组

视频讲解：光盘\TM\lx\7\声明数组.exe

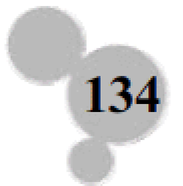
在 PHP 中声明数组的方式主要有两种：一种是应用 array()函数声明数组，另一种是直接通过为数组元素赋值的方式声明数组。其中，应用 array()函数声明数组的方式如下：

```
array array ( [mixed ...])
```

参数 mixed 的语法为 key => value，多个参数 mixed 间用逗号分开，分别定义了索引和值。索引可以是字符串或数字。如果省略了索引，则会自动产生从 0 开始的整数索引。如果索引是整数，则下一个产生的索引将是目前最大的整数索引+1。如果定义了两个完全一样的索引，则后面一个会覆盖前一个。数组中的各数据元素的数据类型可以不同，也可以是数组类型。当 mixed 是数组类型时，就是二维数组（关于二维数组的声明将在 7.5.2 节中进行介绍）。

应用 array()函数声明数组时，数组下标既可以是数值索引也可以是关联索引。下标与数组元素值之间用“=>”进行连接，不同数组元素之间用逗号进行分割。

应用 array()函数定义数组比较灵活，可以在函数体中只给出数组元素值，而不必给出键值。例如：



```
<?php
$array = array ("asp", "php", "jsp");                        //定义数组
print_r($array);                                             //输出数组元素
?>
```

结果为：Array ([0] => asp [1] => php [2] => jsp)

注意

可以通过给变量赋予一个没有参数的 array()函数来创建空数组，然后使用方括号[]语法来添加值。

PHP 提供创建数组的 array()语言结构。在使用其中的数据时，可以直接利用它们在数组中的排列顺序取值，这个顺序称为数组的下标。

```
<?php
echo $array[ 1 ];                                            //输出数组元素的第二个下标值
?>
```

结果为：php

注意

使用这种方式定义数组时，下标默认从 0 开始，而不是 1，然后依次增加 1。所以下标为 2 的元素是指数组的第 3 个元素。

【例 7.1】 下面将通过 array()函数声明数组，实例代码如下：（**实例位置：光盘\TM\sl\7\1**）

```
<?php
$array=array("1"=>"编","2"=>"程","3"=>"词","4"=>"典");      //声明数组
print_r($array);                                             //输出数组元素
echo "<br>";
echo $array[1];                                              //输出数组元素的值
echo $array[2];                                              //输出数组元素的值
echo $array[3];                                              //输出数组元素的值
echo $array[4];                                              //输出数组元素的值
?>
```

结果为：

```
Array ( [1] => 编 [2] => 程 [3] => 词 [4] => 典 )
编程词典
```

PHP 中另一种比较灵活的数组声明方式是直接为数组元素赋值。如果在创建数组时不知道所创建数组的大小，或在实际编写程序时数组的大小可能发生改变，采用这种数组创建的方法较好。

【例 7.2】 为了加深读者对这种数组声明方式的理解，下面通过具体实例对该种数组声明方式进行讲解，实例代码如下：（**实例位置：光盘\TM\sl\7\2**）

```
<?php
$array[1]="编";
```

```
$array[2]="程";
$array[3]="词";
$array[4]="典";
print_r($array);                                    //输出所创建数组的结构
?>
```

结果为：Array ([1] => 编 [2] => 程 [3] => 词 [4] => 典)


注意

通过直接为数组元素赋值方式声明数组时，要求同一数组元素中的数组名相同。

7.3 数组的类型

视频讲解：光盘\TM\lx\7\数组的类型.exe

PHP 支持两种数组：索引数组（indexed array）和联合数组（associative array），前者使用数字作为键，后者使用字符串作为键。

7.3.1 数字索引数组

PHP 数字索引一般表示数组元素在数组中的位置，它由数字组成，下标从 0 开始，数字索引数组默认索引值从数字 0 开始，不需要特别指定，PHP 会自动为索引数组的键名赋一个整数值，然后从这个值开始自动增量，当然，也可以指定从某个位置开始保存数据。

数组可以构造成一系列“键-值（key-value）”对，其中每一对都是数组的一个项目或元素（element）。对于列表中的每个项目，都有一个与之关联的键（key）或索引（index），如表 7.1 所示。

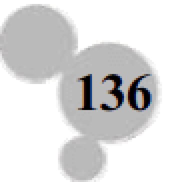
表 7.1 数字索引键值

键	值
0	Low
1	Aimee Mann
2	Ani DiFranco
3	Spiritualized
4	Air

例 7.1 就是一个数字索引数组。

7.3.2 关联数组

关联数组的键名可以是数值和字符串混合的形式，而不像数字索引数组的键名只能为数字，在一



个数组中，只要键名中有一个不是数字，那么这个数组就称为关联数组。

关联数组（associative array）使用字符串索引（或键）来访问存储在数组中的值，如表 7.2 所示。关联索引的数组对于数据库层交互非常有用。

表 7.2　关联数组键值

键	值
MD	Maryland
PA	Pennsylvania
IL	Illinois
MO	Missouri
IA	Iowa

【例 7.3】 本实例将创建一个关联数组，实例代码如下：（**实例位置：光盘\TM\sl\7\3**）

```
<?php
$newarray = array("first"=>1,"second"=>2,"third"=>3);
echo $newarray["second"];
$newarray["third"]=8;
echo $newarray["third"];
?>
```

结果为：28



技巧

关联数组的键名可以是任何一个整数或字符串。如果键名是一个字符串，则不要忘了给这个键名或索引加上一个定界修饰符——单引号（'）或双引号（"）。对于数字索引数组，为了避免不必要的麻烦，最好也加上定界符。

7.4　输 出 数 组

视频讲解：光盘\TM\lx\7\输出数组.exe

在 PHP 中对数组元素进行输出，可以通过输出语句来实现，如 echo 语句、print 语句等，但使用这种输出方式只能对数组中某一元素进行输出。而通过 print_r()函数可以将数组结构进行输出。

语法格式如下：

```
bool print_r ( mixed expression )
```

如果该函数的参数 expression 为普通的整型、字符型或实型变量，则输出该变量本身。如果该参数为数组，则按一定键值和元素的顺序显示出该数组中的所有元素。

【例 7.4】 下面通过一个简单的实例来讲解应用 print_r()函数输出数组的方法，实例代码如下：（实例位置：光盘\TM\sl\7\4）

```
<?php
 $array=array(1=>"PHP5",2=>"从入门",3=>"到精通");
 print_r($array);
?>
```

结果为：Array([1] => PHP5 [2] => 从入门 [3] => 到精通)

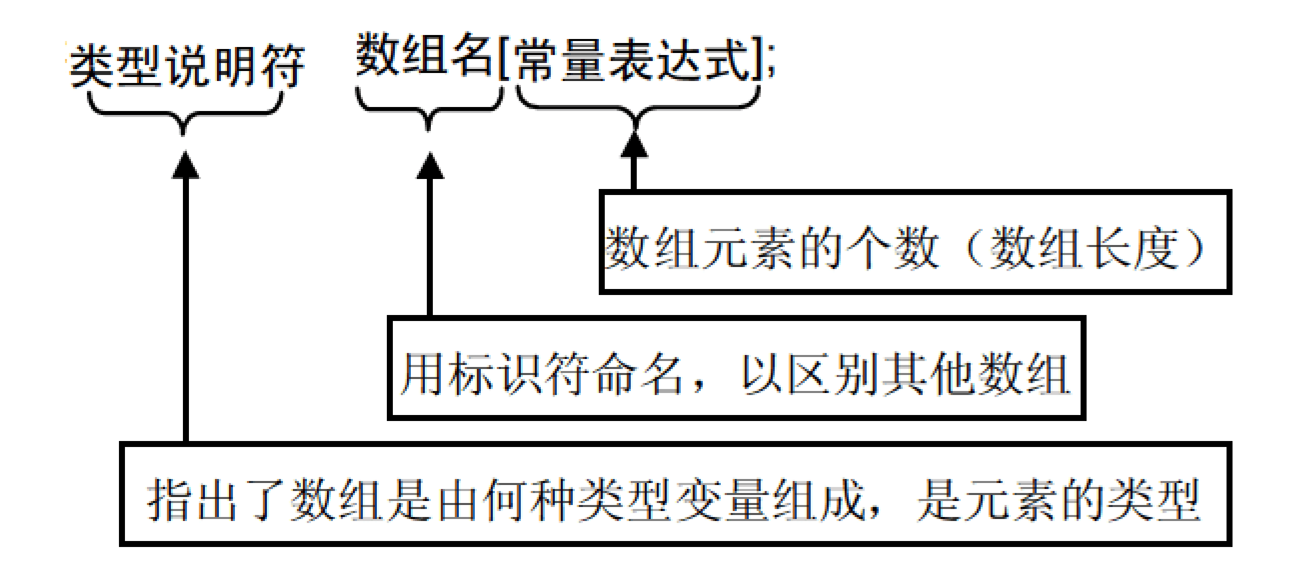
7.5 数组的构造

视频讲解：光盘\TM\lx\7\数组的构造.exe

7.5.1 一维数组

当一个数组的元素是变量时，称这个数组为一维数组。一维数组是最普通的数组，它只保存一列内容。

声明一维数组的一般形式是：



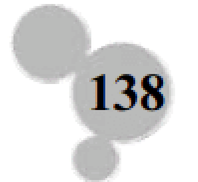
例 7.4 就实现了声明一个一维数组。

7.5.2 二维数组

一个数组的元素如果是一维数组，则称这个数组是二维数组。

【例 7.5】 下面使用具体的实例来声明一个二维数组，实例代码如下：（实例位置：光盘\TM\sl\7\5）

```
<?php
$str = array (
       "书籍"=>array ("文学","历史","地理"),
        "体育用品"=>array ("m"=>"足球","n"=>"篮球"),
        "水果"=>array ("橙子",8=>"葡萄","苹果") );                    //声明数组
```



```
print_r ( $str) ;                                        //输出数组元素
?>
```

结果为：

```
Array(
    [书籍] => Array(
            [0] => 文学
            [1] => 历史
            [2] => 地理
        )
    [体育用品] => Array (
            [m] => 足球
            [n] => 篮球
        )
    [水果] => Array (
            [0] => 橙子
            [8] => 葡萄
            [9] => 苹果
        )
)
```

上面的代码实现了一个二维数组的声明，按照同样的思路，可以创建更高维数的数组，如三维数组。

7.6 遍 历 数 组

视频讲解：光盘\TM\lx\7\遍历数组.exe

遍历数组中的所有元素是常用的一种操作，在遍历的过程中可以完成查询等功能。在生活中，如果想要去商场买一件衣服，就需要在商场中逛一遍，看是否有想要的衣服，逛商场的过程就相当于遍历数组的操作。在 PHP 中遍历数组的方法有多种，下面介绍最常用的两种方法。

1．使用 foreach 结构遍历数组

遍历数组元素最常用的方法是使用 foreach 结构。foreach 结构并非操作数组本身，而是操作数组的一个备份。

【例 7.6】 对于一个存有大量网址的数组变量$url，如果应用 echo 语句一个个地输出，将相当繁琐，而通过 foreach 结构遍历数组则可轻松获取数据信息，实例代码如下：（**实例位置：光盘\TM\sl\7\6**）

```
<?php
 $url = array('编程词典网'=>'www.mrbccd.com',
            '编程体验网'=>'www.bcty365.com',
            '编程资源网'=>'www.bc110.com',
            );                                  //声明数组
foreach (   $url as $link ) {                  //遍历数组
```

```
    echo $link.'<br><br>';
}
?>
```

结果为：www.mrbccd.com www.bcty365.com www.bc110.com

在上面的代码中，PHP 为$url 的每个元素依次执行循环体（each 语句）一次，将$link 赋值给当前元素的值。各元素按数组内部顺序进行处理。

2．使用 list()函数遍历数组

把数组中的值赋给一些变量。与 array()函数类似，这不是真正的函数，而是语言结构。list()函数仅能用于数字索引的数组，且数字索引从 0 开始。

语法格式如下：

```
void list ( mixed ...)
```

参数 mixed 为被赋值的变量名称。

【例 7.7】 下面通过具体的实例讲解 list()函数和 each()函数的综合应用，获取存储在数组中的用户登录信息。**（实例位置：光盘\TM\sl\7\7）**

具体开发步骤如下：

（1）利用开发工具（如 Dreamweaver），新建一个 PHP 动态页，保存为 index.php。

（2）应用 HTML 标记设计页面。首先建立用户登录表单，用于实现用户登录信息的录入，然后使用 each()函数提取全局数组$_POST 中的内容，最后使用 while 语句循环输出用户所提交的注册信息。代码如下：

```
<!--  ---------------------------------------------------定义用户登录表单------------------------------------------------- -->
<form name="form1" method="post">
     <table width="323" border="1" cellpadding="1" cellspacing="1" bordercolor="#66CC33" bgcolor="#FFFFFF">
        <tr>
          <td width="118" height="24" align="right" bgcolor="#CCFF33">用户名：</td>
          <td width="192" height="24" bgcolor="#CCFF33"><input name="user" type="text" class="inputcss" id="user" size="24"></td>
        </tr>
        <tr>
          <td height="24" align="right" bgcolor="#CCFF33">密  码：</td>
          <td height="24" bgcolor="#CCFF33"><input name="pwd" type="password" class="inputcss" id="pwd" size="24"></td>
        </tr>
        <tr align="center" bgcolor="#CCFF33">
          <td height="24" colspan="2"><input name="submit" type="submit"   value="登录"></td>
        </tr>
  </table>
</form>
<?php
//输出用户登录信息
while(list($name,$value)=each($_POST)){
     if($name!="submit"){
```

```
            echo "$name=$value<br>";
        }
}
?>
```

（3）在 IE 浏览器中输入地址，按 Enter 键，输入用户名及密码，单击“登录”按钮，运行结果如图 7.2 所示。



图 7.2　应用 list()函数获取用户登录信息


说明

each()函数用于返回当前指针位置的数组值，并将指针推进一个位置。返回的数组包含 4 个键，键 0 和 key 包含键名，而键 1 和 value 包含相应的数据。如果程序在执行 each()函数时指针已经位于数组末尾，则返回 false。

7.7　字符串与数组的转换

视频讲解：光盘\TM\lx\7\字符串与数组的转换.exe

字符串与数组的转换在程序开发过程中经常使用，主要使用 explode()函数和 implode()函数实现，下面分别进行详细讲解。

1．使用 explode()函数将字符串转换成数组

explode()函数将字符串依指定的字符串或字符 separator 切开。

语法格式如下：

```
array explode(string separator, string string, [int limit])
```

返回由字符串组成的数组，每个元素都是 string 的一个子串，它们被字符串 separator 作为边界点分隔出来。如果设置了 limit 参数，则返回的数组包含最多 limit 个元素，而最后那个元素将包含 string 的剩余部分；如果 separator 为空字符串（""），explode()函数将返回 false；如果 separator 所包含的值在 string 中找不到，那么 explode()函数将返回包含 string 单个元素的数组；如果参数 limit 是负数，则

返回除了最后的-limit 个元素外的所有元素。

【例 7.8】 本实例使用 explode()函数将“时装、休闲、职业装”字符串按照“、”进行分隔，实例代码如下：（**实例位置：光盘\TM\sl\7\8**）

```
<?php
$str = "时装、休闲、职业装";                    //定义一个字符串
$strs = explode("、", $str);                    //应用 explode()函数将字符串转换成数组
print_r($strs);                                 //输出数组元素
?>
```

结果为：Array ([0] => 时装 [1] => 休闲 [2] => 职业装)

【例 7.9】 在开发一个投票管理系统时，经常需要在后台添加投票选项到投票系统，以作为投票的内容。下面使用 explode()函数对添加的投票选项通过“*”进行区分，然后使用 while 循环语句分别在页面中输出添加的投票选项。（**实例位置：光盘\TM\sl\7\9**）

具体开发步骤如下：

（1）利用开发工具（如 Dreamweaver），新建一个 PHP 动态页，保存为 index.php。

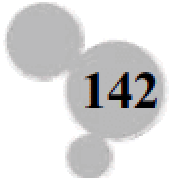
（2）使用 HTML 标记设计页面，首先建立投票表单，用于实现添加投票选项，然后使用 each()函数提取全局数组$_POST 中的内容，并最终使用 while 循环输出投票选项内容。代码如下：

```
<!-- ---------------------------------------------------定义添加投票表单--------------------------------------------------- -->
<form name="form1" method="post" action="">
  <table width="400" border="1" cellpadding="0" cellspacing="1" bordercolor="#FF9900" bgcolor="#CCFF66">
        <tr align="center">
          <td width="98" height="120">添加投票选项：</td>
          <td width="223" height="120"><p>
              <textarea name="content" cols="30" rows="5" id="content"></textarea>
              <br>
              <span class="style1">注意：每个选项间用*分隔</span></p></td>
          <td width="61" height="120"><input type="Submit" name="Submit" value="提交"></td>
        </tr>
      </table>
    </form>
```

（3）添加一个表格，然后在表格的单元格中添加以下代码，用来输出添加的投票选项。

```
<?php
if($_POST[Submit]!=""){
     $content=$_POST[content];
     $data=explode("*",$content);
     while(list($name,$value)=each($data)){
          echo '<input type="checkbox" name="checkbox" value="checkbox">';
          echo $value."\n";
     }
  }
?>
```

（4）在 IE 浏览器中输入地址，按 Enter 键，输入投票选项的内容，各选项间用“*”进行分隔，



单击“提交”按钮，运行结果如图 7.3 所示。



图 7.3　在投票系统的后台管理中使用 explode()函数

2．使用 implode()函数将数组转换成一个新字符串

将数组的内容组合成一个字符串。

语法格式如下：

```
string implode(string glue, array pieces)
```

参数 glue 是字符串类型，指要传入的分隔符；参数 pieces 是数组类型，指传入的要合并元素的数组变量名称。

【例 7.10】 使用 implode()函数将数组中的内容以空格作分隔符进行连接，从而组合成一个新的字符串，实例代码如下：**（实例位置：光盘\TM\sl\7\10）**

```
<?php
$str=array(明日,编程词典,网址,'www.mrbccd.com',服务电话,"0431-84972266");
echo implode(" ",$str);                          //以空格作为分隔符将数组中的元素组合成一个新字符串
?>
```

结果为：明日　编程词典　网址　www.mrbccd.com　服务电话　0431-84972266

7.8　统计数组元素个数

视频讲解：光盘\TM\lx\7\统计数组元素的个数.exe

在 PHP 中，使用 count()函数对数组中的元素个数进行统计。

语法格式如下：

```
int count ( mixed array [, int mode])
```

count()函数的参数说明如表 7.3 所示。

表 7.3　count()函数的参数说明

参　数	说　明
array	必要参数。输入的数组
mode	可选参数。COUNT_RECURSIVE（或 1），如选中此参数，本函数将递归地对数组计数。对计算多维数组的所有单元尤其有用。此参数的默认值为 0

例如，使用 count()函数统计数组元素的个数，实例代码如下：

```
<?php
$array = array("PHP 函数参考大全","PHP 程序开发范例宝典","PHP 网络编程自学手册","PHP5 从入门到精通 ");
echo count($array);                                    //统计数组元素的个数，输出结果为 4
?>
```

【例 7.11】 将图书的数据存放在数组中，使用 count()函数递归地统计数组中图书数量并输出，实例代码如下：（**实例位置：光盘\TM\sl\7\11**）

```
<?php
$array = array("php" => array("PHP 函数参考大全","PHP 程序开发范例宝典","PHP 数据库系统开发完全手册"),
                    "asp" => array("ASP 经验技巧宝典")
            );                                                          //声明一个二维数组
echo count($array,COUNT_RECURSIVE);                                     //递归统计数组元素的个数
?>
```

结果为：6

注意

在统计二维数组时，如果直接使用 count()函数只会显示到一维数组的个数，所以使用递归的方式来统计二维数组的个数。

7.9　查询数组中指定元素

视频讲解：光盘\TM\lx\7\查询数组中指定元素.exe

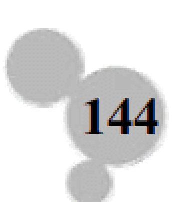
array_search()函数，在数组中搜索给定的值，找到后返回键名，否则返回 false。在 PHP 4.2.0 之前，函数在失败时返回 null 而不是 false。

语法格式如下：

```
mixed array_search ( mixed needle, array haystack [, bool strict])
```

参数 needle 指定在数组中搜索的值；参数 haystack 指定被搜索的数组；参数 strict 为可选参数，如果值为 true，还将在数组中检查给定值的类型。

【例 7.12】 本实例综合应用数组函数，实现更新数组中元素的值，实例代码如下：（**实例位置：**



光盘\TM\sl\7\12）

```
<?php
$name = "智能机器人@数码相机@天翼 3G 手机@瑞士手表";          //定义字符串
$price ="14998@2588@2666@66698";
$counts = "1@2@3@4";
$arrayid=explode("@",$name);                     //将商品 ID 的字符串转换到数组中
$arraynum=explode("@",$price);                   //将商品数量的字符串转换到数组中
$arraycount=explode("@",$counts);                //将商品数量的字符串转换到数组中
if($_POST[Submit]==true){
$id=$_POST[name];                                //获取要更改的元素名称
$num=$_POST[counts];                             //获取更改的值
$key=array_search($id,$arrayid);                 //在数组中搜索给定的值，如果成功返回键名
$arraycount[$key]=$num;                          //更改商品数量
$counts=implode("@",$arraycount);                //将更改后的商品数量添加到购物车中
}
?>
<table width="580" border="1" cellpadding="1" cellspacing="1" bordercolor="#FFFFFF" bgcolor="#c17e50">
    <tr>
        <td width="145" align="center" bgcolor="#FFFFFF"   class="STYLE1">商品名称</td>
        <td width="145" align="center" bgcolor="#FFFFFF"   class="STYLE1">价  格</td>
        <td width="145" align="center" bgcolor="#FFFFFF"   class="STYLE1">数量</td>
        <td width="145" align="center" bgcolor="#FFFFFF"   class="STYLE1">金额</td>
    </tr>
    <?php
    for($i=0;$i<count($arrayid);$i++){                   //for 循环读取数组中的数据
    ?>
    <form name="form1_<?php echo $i;?>" method="post" action="index.php">
    <tr>
        <td height="25" align="center" bgcolor="#FFFFFF" class="STYLE2"><?php echo $arrayid[$i]; ?></td>
        <td align="center" bgcolor="#FFFFFF" class="STYLE2"><?php echo $arraynum[$i]; ?></td>
        <td align="center" bgcolor="#FFFFFF" class="STYLE2">
            <input name="counts" type="text" id="counts" value="<?php echo $arraycount[$i]; ?>" size="8">
            <input name="name" type="hidden" id="name" value="<?php echo $arrayid[$i]; ?>">
            <input type="submit" name="Submit" value="更改"></td>
        <td align="center" class="STYLE2"><?php echo $arraycount[$i]*$arraynum[$i]; ?></td>
    </tr>
    </form>
    <?php
    }
    ?>
</table>
```

在本实例中，实现对数组中存储的商品数量进行修改，其运行结果如图 7.4 所示。

说明

array_search()函数最常见的应用是购物车，实现对购物车中指定的商品数量的修改和删除。



图 7.4　更新数组中元素的值

7.10　获取数组中最后一个元素

视频讲解：光盘\TM\lx\7\获取数组中最后一个元素.exe

通过函数 array_pop()获取数组中的最后一个单元。

array_pop()函数获取并返回数组的最后一个单元，并将数组的长度减 1，如果数组为空（或者不是数组）将返回 null。

语法格式如下：

```
mixed array_pop ( array array)
```

参数 array 为输入的数组。

【例 7.13】 本实例应用 array_pop()函数获取数组中的最后一个元素，实例代码如下：（**实例位置：光盘\TM\sl\7\13**）

```
<?php
$arr = array ("asp", "java", "javaweb", "php", "vb");              //定义数组
$array = array_pop ($arr);                                        //获取数组中最后一个元素
echo "被弹出的单元是：$array <br />";                              //输出最后一个元素值
print_r($arr);                                                    //输出数组结构
?>
```

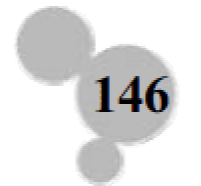
结果为：被弹出的单元是：vb Array ([0] => asp [1] => java [2] => javaweb [3] => php)

7.11　向数组中添加元素

视频讲解：光盘\TM\lx\7\向数组中添加元素.exe

通过 array_push()函数向数组中添加元素。

array_push()函数将数组当成一个栈，将传入的变量压入该数组的末尾，该数组的长度将增加入栈



变量的数目，返回数组新的单元总数。

语法格式如下：

```
int array_push ( array array, mixed var [, mixed ...])
```

参数 array 为指定的数组，参数 var 是压入数组中的值。

【例 7.14】 本实例应用 array_push()函数向数组中添加元素，实例代码如下：**（实例位置：光盘\TM\sl\7\14）**

```
<?php
$array_push = array ("PHP 从入门到精通", "PHP 范例手册");                    //定义数组
array_push ($array_push, "PHP 开发典型模块大全","PHP 网络编程自学手册");      //添加元素
print_r($array_push);              //输出数组结果
?>
```

结果为：

```
Array ( [0] => PHP 从入门到精通 [1] => PHP 范例手册 [2] => PHP 开发典型模块大全 [3] => PHP 网络编程自学
手册 )
```

7.12　删除数组中重复元素

视频讲解：光盘\TM\lx\7\删除数组中重复元素.exe

通过 array_unique()函数删除数组中重复的元素。

array_unique()函数，将值作为字符串排序，然后对每个值只保留第一个键名，忽略所有后面的键名，即删除数组中重复的元素。

语法格式如下：

```
array array_unique ( array array)
```

参数 array 为输入的数组。

【例 7.15】 本实例应用 array_unique()函数删除数组中重复的元素，实例代码如下：**（实例位置：光盘\TM\sl\7\15）**

```
<?php
$array_push = array ("PHP 从入门到精通", "PHP 范例手册", "PHP 范例手册","PHP 网络编程自学手册");
                                        //定义数组
array_push ($array_push, "PHP 开发典型模块大全","PHP 网络编程自学手册");
                                        //添加元素
print_r($array_push);                   //输出数组
echo "<br>";
$result=array_unique($array_push);      //删除数组中重复的元素
print_r($result);                       //输出删除后的数组
?>
```

结果为：

```
Array ( [0] => PHP 从入门到精通 [1] => PHP 范例手册 [2] => PHP 范例手册 [3] => PHP 网络编程自学手册 [4] => PHP 开发典型模块大全 [5] => PHP 网络编程自学手册 )
Array ( [0] => PHP 从入门到精通 [1] => PHP 范例手册 [3] => PHP 网络编程自学手册 [4] => PHP 开发典型模块大全 )
```

7.13 综合运用数组函数-实现多文件上传

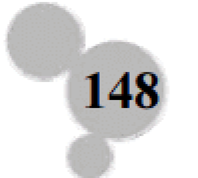
【例 7.16】 本实例综合运用数组函数，实现同时将任意多个文件上传到服务器的功能。这里文件的上传使用的是 move_uploaded_file()函数，使用 array_push()函数向数组中添加元素，使用 array_unique()函数删除数组中重复元素，使用 array_pop()函数获取数组中最后一个元素，并将数组长度减 1，使用 count()函数获取数组的元素个数。实例代码如下：**（实例位置：光盘\TM\sl\7\16）**

首先，在 index.php 文件中创建表单，指定使用 post 方法提交数据，设置 enctype="multipart/form-data" 属性，添加表单元素，完成文件的提交操作。

```
<form action="index_ok.php" method="post" enctype="multipart/form-data" name="form1">
          <tr>
            <td width="88" height="30" align="right" class="STYLE1">内容 1：</td>
            <td width="369"><input name="picture[]" type="file" id="picture[]" size="30"></td>
            </tr>
        ……//省略了部分代码
          <tr>
            <td height="30" align="right" class="STYLE1">内容 5：</td>
            <td><input name="picture[]" type="file" id="picture[]" size="30"></td>
            </tr>
          <tr>
            <td><input type="image" name="imageField" src="images/02-03 (3).jpg"></td>
            </tr>
       </form>
```

然后，在 index_ok.php 文件中，通过$_FILES 预定义变量获取表单提交的数据，通过数组函数完成对上传文件元素的计算，最后使用 move_uploaded_file()函数将上传文件添加到服务器指定文件夹下。

```
<?php
     if(!is_dir("./upfile")){                              //判断服务器中是否存在指定文件夹
          mkdir("./upfile");                               //如果不存在，则创建文件夹
     }
     array_push($_FILES["picture"]["name"],"");            //向表单提交的数组中增加一个空元素
    $array=array_unique($_FILES["picture"]["name"]);       //删除数组中重复的值
    array_pop($array);                                     //删除数组中最后一个单元
     for($i=0;$i<count($array);$i++){                      //根据元素个数执行 for 循环
     $path="upfile/".$_FILES["picture"]["name"][$i];       //定义上传文件存储位置
          if(move_uploaded_file($_FILES["picture"]["tmp_name"][$i],$path)){   //执行文件上传操作
```



```
                $result=true;
            }else{
                $result=false;
            }
        }
        if($result==true){
                echo "文件上传成功，请稍等...";
                echo "<meta http-equiv=\"refresh\" content=\"3; url=index.php\">";
            }else{
                echo "文件上传失败，请稍等...";
                echo "<meta http-equiv=\"refresh\" content=\"3; url=index.php\">";
        }
?>
```

运行结果如图 7.5 所示。



图 7.5　在多文件上传中应用数组函数

注意

通过 POST 方法实现多文件上传，在创建 form 表单时，必须指定 enctype="multipart/form-data" 属性。

7.14　小　　结

本章的重点是数组的常用操作，这些操作会在实际应用中经常用到。另外，PHP 提供了大量的数组函数，完全可以在开发任务中轻松实现所需要的功能。希望通过本章的学习，读者能够举一反三，

对所学知识进行灵活运用，开发实用的 PHP 程序。

7.15 练习与实践

1. 尝试声明一个一维数组和一个二维数组，并对数组元素进行输出。**（答案位置：光盘\TM\sl\7\17）**

2. 尝试开发一个页面，使用 list()函数和 each()函数获取存储在数组中的图书名称和作者。**（答案位置：光盘\TM\sl\7\18）**

3. 尝试开发一个页面，使用 explode()函数以“*”为分隔符实现添加多选题功能。**（答案位置：光盘\TM\sl\7\19）**

4. 尝试开发一个页面，使用 sort()函数对指定的数组进行升序排序。**（答案位置：光盘\TM\sl\7\20）**

第 8 章

PHP 与 Web 页面交互

（视频讲解：1 小时 1 分钟）

PHP 与 Web 页面交互是学习 PHP 语言编程的基础。在 PHP 中提供了两种与 Web 页面交互的方法，一种是通过 Web 表单提交数据，另一种是通过 URL 参数传递。

本章将详细讲解 PHP 与 Web 页面交互的相关知识，为以后学习 PHP 语言编程做好铺垫。

通过阅读本章，您可以：

- 了解表单及表单元素
- 熟悉在 Web 页中插入表单的过程
- 了解获取表单数据的两种方法
- 掌握 PHP 传递参数的常用方法
- 掌握对 URL 传递参数编码和解码技术
- 掌握如何在 Web 页中插入 PHP 脚本
- 熟练掌握获取各种表单数据的操作
- 掌握 PHP 与 Web 表单的综合应用

8.1　表　　单

视频讲解：光盘\TM\lx\8\Web 页表单元素的组成.exe

Web 表单的功能是让浏览者和网站有一个互动的平台。Web 表单主要用来在网页中发送数据到服务器，例如，提交注册信息时需要使用表单。当用户填写完信息后做提交（submit）操作，于是将表单的内容从客户端的浏览器传送到服务器端，经过服务器上的 PHP 程序进行处理后，再将用户所需要的信息传递回客户端的浏览器上，从而获得用户信息，使 PHP 与 Web 表单实现交互。

8.1.1　创建表单

使用<form>元素，并在其中插入相关的表单元素，即可创建一个表单。

表单结构：

```
<form name="form_name"   method="method" action="url"   enctype="value"   target="target_win">
…                                 //省略插入的表单元素
</form >
```

<form>标记的属性如表 8.1 所示。

表 8.1　<form>标记的属性

<form>标记的属性	说　　明
name	表单的名称
method	设置表单的提交方式，GET 或者 POST 方法
action	指向处理该表单页面的 URL（相对位置或者绝对位置）
enctype	设置表单内容的编码方式
target	设置返回信息的显示方式，target 的属性值如表 8.2 所示

表 8.2　target 属性值

属　性　值	描　　述
_blank	将返回信息显示在新的窗口中
_parent	将返回信息显示在父级窗口中
_self	将返回信息显示在当前窗口中
_top	将返回信息显示在顶级窗口中

说明

GET 方法是将表单内容附加在 URL 地址后面发送；POST 方法是将表单中的信息作为一个数据块发送到服务器上的处理程序中，在浏览器的地址栏不显示提交的信息。method 属性默认方法为 GET 方法。

例如，创建一个表单，再以 POST 方法提交到数据处理页 check_ok.php，代码如下：

```
<form name="form1" method="post" action="check_ok.php">
</form>
```

以上代码中的<form>标记的属性是最基本的使用方法。需要注意的是，在使用 form 表单时，必须指定其行为属性 action，它指定表单在提交时将内容发往何处进行处理。

8.1.2　表单元素

表单（form）由表单元素组成。常用的表单元素有以下几种标记：输入域标记<input>、选择域标记<select>和<option>、文字域标记<textarea>等。

1．输入域标记<input>

输入域标记<input>是表单中最常用的标记之一。常用的文本框、按钮、单选按钮、复选框等构成了一个完整的表单。

语法格式如下：

```
<form>
<input name="file_name"   type="type_name">
</form>
```

参数 name 是指输入域的名称，参数 type 是指输入域的类型。在<input type="">标记中一共提供了 10 种类型的输入区域，用户所选择使用的类型由 type 属性决定。type 属性取值及举例如表 8.3 所示。

表 8.3　type 属性取值及举例

值	举　　例	说　　明	运 行 结 果
text	<input name="user" **type="text"** value="纯净水" size="12" maxlength="1000">	name 为文本框的名称，value 是文本框的默认值，size 指文本框的宽度（以字符为单位），maxlength 指文本框的最大输入字符数	添加一个文本框：纯净水
password	<input name="pwd" **type="password"**value="666666" size="12" maxlength="20">	密码域，用户在该文本框中输入的字符将被替换显示为*，以起到保密作用	添加一个密码域：******
file	<input name="file" **type="file"enctype="multipart/form-data"**size="16" maxlength="200">	文件域，当文件上传时，可用来打开一个模式窗口以选择文件。然后将文件通过表单上传到服务器，如上传 word 文件等。必须注意的是，上传文件时需要指明表单的属性 enctype="multipart/form-data"才可以实现上传功能	添加一个文件域：浏览...
image	<input name="imageField" **type="image"** src="images/banner.gif" width="120"height="24" border="0">	图像域是指可以用在提交按钮位置上的图片，这幅图片具有按钮的功能	添加一个图像域：

续表

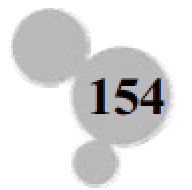
值	举　　例	说　　明	运行结果
radio	<input name="sex" **type="radio"** value="1" **checked**>男 <input name="sex" **type="radio"** value="0">女	单选按钮，用于设置一组选择项，用户只能选择一项。checked 属性用来设置该单选按钮默认被选中	添加一组单选按钮（例如，您的性别为：） ◉ 男 ○ 女
checkbox	<input name="checkbox" **type="checkbox"** value="1" **checked**> 封面 <input name="checkbox" **type="checkbox"** value="1" **checked**> 正文内容 <input name="checkbox" **type="checkbox"** value="0">价 格	复选框，允许用户选择多个选择项。checked 属性用来设置该复选框默认被选中。例如，收集个人信息时，要求在个人爱好的选项中进行多项选择等	添加一组复选框，（如影响您购买本书的因素：） ☑ 封面 ☑ 正文内容 ☐ 价 格
submit	<input **type="submit"**name="Submit"value="提交">	将表单的内容提交到服务器端	添加一个提交按钮： 提交
reset	<input **type="reset"** name="Submit" value="重置">	清除与重置表单内容，用于清除表单中所有文本框的内容，并使选择菜单项恢复到初始值	添加一个重置按钮： 重置
button	<input **type="button"** name="Submit" value="按钮">	按钮可以激发提交表单的动作，可以在用户需要修改表单时，将表单恢复到初始的状态，还可以依照程序的需要发挥其他作用。普通按钮一般是配合 JavaScript 脚本进行表单处理的	添加一个普通按钮： 按钮
hidden	<input **type="hidden"** name="bookid">	隐藏域，用于在表单中以隐含方式提交变量值。隐藏域在页面中对于用户是不可见的，添加隐藏域的目的在于通过隐藏的方式收集或者发送信息。浏览者单击“发送”按钮发送表单时，隐藏域的信息也被一起发送到 action 指定的处理页	添加一个隐藏域：

2. 选择域标记<select>和<option>

通过选择域标记<select>和<option>可以建立一个列表或者菜单。菜单的使用是为了节省空间，正常状态下只能看到一个选项，单击右侧的下三角按钮打开菜单后才能看到全部的选项。列表可以显示一定数量的选项，如果超出了这个数量，会自动出现滚动条，浏览者可以通过拖动滚动条来查看各选项。

语法格式如下：

```
<select name="name" size="value" multiple>
<option value="value" selected>选项 1</option>
<option value="value">选项 2</option>
```



```
<option value="value">选项 3</option>
…
</select>
```

参数 name 表示选择域的名称；参数 size 表示列表的行数；参数 value 表示菜单选项值；参数 multiple 表示以菜单方式显示数据，省略则以列表方式显示数据。

选择域标记<select>和<option>的显示方式及举例如表 8.4 所示。

表 8.4　选择域标记<select>和<option>的显示方式及举例

显示方式	举　例	说　明	运行结果
列表方式	<**select** name="spec" id="spec"> <**option** value="0" **selected**>网络编程</**option**> <**option** value="1">办公自动化</**option**> <**option** value="2">网页设计</**option**> <**option** value="3">网页美工</**option**> </**select**>	下拉列表框，通过选择域标记<select>和<option>建立一个列表，列表可以显示一定数量的选项，如果超出了这个数量，会自动出现滚动条，浏览者可以通过拖动滚动条来查看各选项。selected 属性用来设置该菜单时默认被选中	请选择所学专业： 网络编程 网络编程 办公自动化 网页设计 网页美工
菜单方式	<**select** name="spec" id="spec" **multiple** > <**option** value="0" **selected**>网络编程</**option**> <**option** value="1">办公自动化</**option**> <**option** value="2">网页设计</**option**> <**option** value="3">网页美工</**option**> </**select**>	multiple 属性用于下拉列表<select>标记中，指定该选项用户可以使用 Shift 和 Ctrl 键进行多选	请选择所学专业： 网络编程 办公自动化 网页设计 网页美工

说明

在上面的表格中给出了静态菜单项的添加方法，而在 Web 程序开发过程中，也可以通过循环语句动态添加菜单项。

3．文字域标记<textarea>

文字域标记<textarea>用来制作多行的文字域，可以在其中输入更多的文本。

语法格式如下：

```
<textarea name="name" rows=value cols=value value="value" warp="value">
    …文本内容
</textarea>
```

参数 name 表示文字域的名称；rows 表示文字域的行数；cols 表示文字域的列数（这里的 rows 和 cols 以字符为单位）；value 表示文字域的默认值，warp 用于设定显示和送出时的换行方式，值为 off 表示不自动换行，值为 hard 表示自动硬回车换行，换行标记一同被发送到服务器，输出时也会换行，值为 soft 表示自动软回车换行，换行标记不会被发送到服务器，输出时仍然为一列。

文字域标记<textarea>的值及举例如表 8.5 所示。

表 8.5　文字域标记<textarea>的值及举例

值	举　例	说　明	运 行 结 果
textarea	<**textarea** name="remark" cols="20" rows="4" id="remark">　请输入您的建议!</**textarea**>	文本域，也称多行文本框，用于多行文本的编辑 warp 属性默认为自动换行方式	请发表您的建议： 请输入您的建议！

【例 8.1】 下面通过具体的实例，了解 warp 属性的 hard 和 soft 换行标记的区别，实例代码如下：（**实例位置：光盘\TM\sl\8\1**）

```
<form name="form1" method="post" action="index.php">
<textarea name="a" cols="20" rows="3" wrap="soft">我使用的是软回车！我输出后不换行！</textarea>
<textarea name="b" cols="20" rows="3" wrap="hard">我使用的是硬回车！我输出后自动换行！</textarea>
<input type="submit" name="Submit" value="提交">
</form>
<?php
echo nl2br($_POST[a])."<br>";
echo nl2br($_POST[b]);
?>
```

HTML 标记在获取多行编辑框中的字符串时，并不会显示换行标记。在上面的代码中使用了 nl2br()函数将换行符“\n”替换成“
”换行标识，并应用 echo 语句进行输出。运行结果如图 8.1 所示。

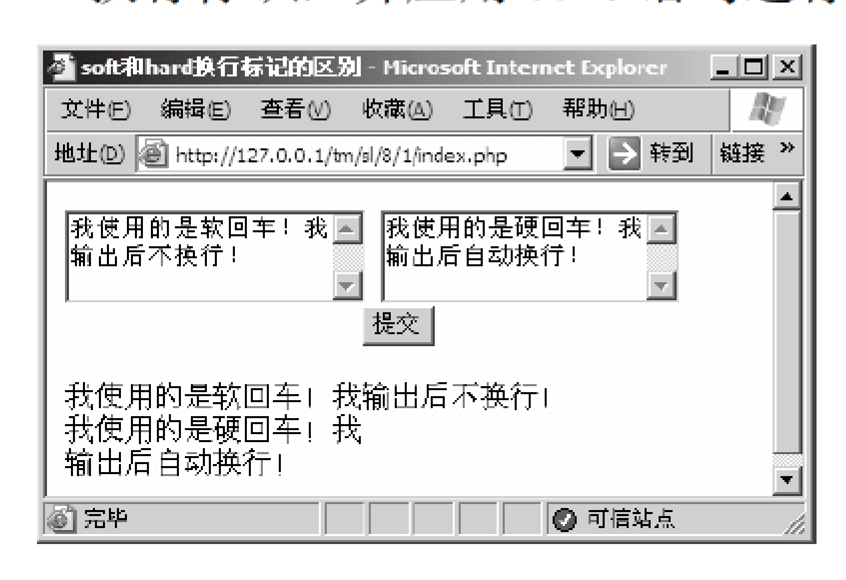


图 8.1　soft 和 hard 换行标记的区别

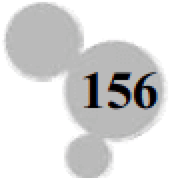
注意

hard 和 soft 换行标记的使用效果在浏览器上是看不出来的，只有在提交表单后选择 IE 浏览器的“查看”/“源文件”命令，才能看出执行换行标记后的效果，或者通过 nl2br()函数进行转换后查看。

8.2　在普通的 Web 页中插入表单

视频讲解：光盘\TM\lx\8\在普通的 Web 页中插入表单.exe

【例 8.2】 在普通的 Web 页中插入表单的操作步骤如下：（**实例位置：光盘\TM\sl\8\2**）



（1）在 HTML 的<body>…</body>标记中添加一个表单。

（2）在表单中添加表单元素，代码如下：

```
<form action="index.php" method="post" name="form1" enctype="multipart/form-data">
  <table width="405" border="1" cellpadding="1" cellspacing="1" bordercolor="#FFFFFF" bgcolor="#999999">
    <tr bgcolor="#FFCC33">
      <td width="103" height="25" align="right">姓名：</td>
      <td width="144" height="25"><input name="user" type="text" id="user" size="20" maxlength="100"></td>
    </tr>
    <tr bgcolor="#FFCC33">
      <td height="25" align="right">性别：</td>
      <td height="25" colspan="2" align="left"><input name="sex" type="radio" value="男" checked>男
        <input type="radio" name="sex" value="女">女</td>
    </tr>
    <tr bgcolor="#FFCC33">
      <td width="103" height="25" align="right">密码：</td>
      <td  width="289"  height="25"  colspan="2"  align="left"><input  name="pwd"  type="password"  id="pwd"
size="20" maxlength="100"></td>
    </tr>
    <tr bgcolor="#FFCC33">
      <td height="25" align="right">学历：</td>
      <td height="25" colspan="2" align="left"><select name="select">
          <option value="专科">专科</option>
          <option value="本科" selected>本科</option>
        </select></td>
    </tr>
    <tr bgcolor="#FFCC33">
      <td height="25" align="right">爱好：</td>
      <td height="25" colspan="2" align="left">
        <input name="fond[]" type="checkbox" id="fond[]" value="音乐">音乐
        <input name="fond[]" type="checkbox" id="fond[]" value="其他">其他
       </td>
    </tr>
    <tr bgcolor="#FFCC33">
      <td height="25" align="right">个人写真：  </td>
      <td height="25" colspan="2"><input name="photo" type="file" size="20" maxlength="1000" id="photo"></td>
    </tr>
    <tr bgcolor="#FFCC33">
      <td height="25" align="right">个人简介：  </td>
      <td height="25" colspan="2"><textarea name="intro" cols="28" rows="4" id="intro"></textarea></td>
    </tr>
    <tr align="center" bgcolor="#FFCC33">
      <td height="25" colspan="3"><input type="submit" name="submit" value="提交">
        <input type="reset" name="submit2" value="重置"></td>
    </tr>
  </table>
</form>
```

（3）将该文件保存为 index.php 页。

注意

由于该页未使用 PHP 脚本，因此该文件属于静态页，可以将其保存为.html 格式，然后直接使用浏览器打开该文件查看运行结果即可。

（4）在 IE 浏览器中输入地址，按 Enter 键，运行结果如图 8.2 所示。

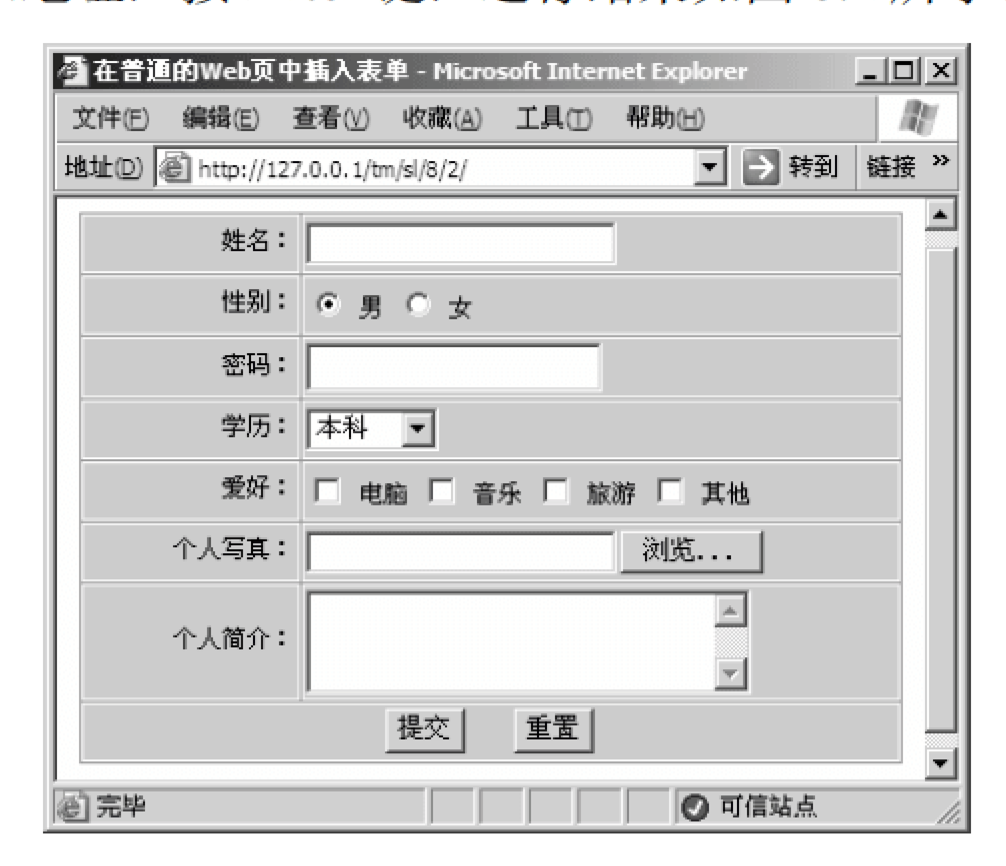


图 8.2　在普通的 Web 页中插入表单

8.3　获取表单数据的两种方法

视频讲解：光盘\TM\lx\8\获取表单数据的两种方法.exe

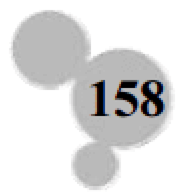
获取表单元素提交的值是表单应用中最基本的操作，表单数据的传递方法有两种，即 POST 方法和 GET 方法。采用哪种方法是由<form>表单的 method 属性所指定的，下面讲解这两种方法在 Web 表单中的应用。

8.3.1　使用 POST 方法提交表单

应用 POST 方法时，只需将<form>表单中的属性 method 设置成 POST 即可。POST 方法不依赖于 URL，不会显示在地址栏。POST 方法可以没有限制地传递数据到服务器，所有提交的信息在后台传输，用户在浏览器端是看不到这一过程的，安全性高。所以 POST 方法比较适合用于发送一个保密的（如信用卡号）或者容量较大的数据到服务器。

【例 8.3】 下面的实例将使用 POST 方法发送文本框信息到服务器，实例代码如下：（**实例位置：光盘\TM\sl\8\3**）

```
<form name="form1" method="post" action="index.php">
<table width="300" border="0" cellpadding="0" cellspacing="0">
```



```
<tr>
<td height="30">  订单号：
  <input type="text" name="user" size="20" >
  <input type="submit" name="submit" value="提交">
</td>
  </tr>
</table>
</form>
```

在上面的代码中，form 表单的 method 属性指定了 POST 方法的传递方式，并通过 action 属性指定了数据处理页为 index.php，因此，当单击“提交”按钮后，即提交文本框的信息到服务器。运行结果如图 8.3 所示。



图 8.3　使用 POST 方法提交表单

8.3.2　使用 GET 方法提交表单

GET 方法是<form>表单中 method 属性的默认方法。使用 GET 方法提交的表单数据被附加到 URL 后，并作为 URL 的一部分发送到服务器端。在程序的开发过程中，由于 GET 方法提交的数据是附加到 URL 上发送的，因此，在 URL 的地址栏中将会显示“URL+用户传递的参数”。

GET 方法的传参格式如下：

```
http://url?name1=value1&name2=value2……
```



其中，url 为表单响应地址（如 127.0.0.1/index.php），name1 为表单元素的名称，value1 为表单元素的值。url 和表单元素之间用“?”隔开，而多个表单元素之间用“&”隔开，每个表单元素的格式都是 name=value，固定不变。

注意

若要使用 GET 方法发送表单，URL 的长度应限制在 1MB 字符以内。如果发送的数据量太大，数据将被截断，从而导致意外或失败的处理结果。

【例 8.4】 下面创建一个表单来实现应用 GET 方法提交用户名和密码，并显示在 URL 地址栏中。添加一个文本框，命名为 user，添加一个密码域，命名为 pwd，将表单的 method 属性设置为 GET 方

法，实例代码如下：**（实例位置：光盘\TM\sl\8\4）**

```
<form name="form1" method="get" action="index.php">
  <table width="352" border="0" cellpadding="0" cellspacing="0">
    <tr>
      <td width="352" height="30">   用户名：
        <input name="user" type="text" size="12" >
        密码：
        <input name="pwd" type="password" id="pwd" size="12">
        <input type="submit" name="submit" value="提交"></td>
    </tr>
  </table>
</form>
```

运行本实例，在文本框中输入用户名和密码，单击“提交”按钮，文本框内的信息就会显示在 URL 地址栏中，如图 8.4 所示。



图 8.4　使用 GET 方法提交表单

显而易见，这种方法会将参数暴露。如果用户传递的参数是非保密性的参数（如 id=8），那么采用 GET 方法传递数据是可行的，如果用户传递的是保密性的参数（如密码），这种方法就会不安全。解决该问题的方法是将表单的 method 属性指定的 GET 方法改为 POST 方法。

8.4　PHP 参数传递的常用方法

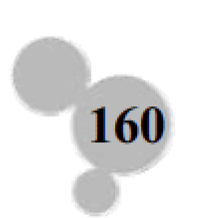
视频讲解：光盘\TM\lx\8\PHP 传参的常用方法.exe

PHP 参数传递常用的方法有 3 种：$_POST[]、$_GET[]、$_SESSION[]，分别用于获取表单、URL 与 Session 变量的值。

8.4.1　$_POST[]全局变量

使用 PHP 的$_POST[]预定义变量可以获取表单元素的值，格式为：

```
$_POST[name]
```



例如，建立一个表单，设置 method 属性为 POST，添加一个文本框，命名为 user，获取表单元素的代码如下：

```
<?php
$user=$_POST["user"];                    //应用$_POST[]全局变量获取表单元素中文本框的值
?>
```

说明

在某些 PHP 版本中直接写$user 即可调用表单元素的值，这和 php.ini 的配置有关系。在 php.ini 文件中检索到 register_globals=ON/OFF 这行代码，如果为 ON，就可以直接写成$user，反之则不可以。虽然直接应用表单名称十分方便，但也存在一定的安全隐患。此处推荐使用 register_globals=OFF。

8.4.2　$_GET[]全局变量

PHP 使用$_GET[]预定义变量获取通过 GET 方法传过来的值，使用格式为：

```
$_GET[name]
```

这样就可以直接使用名字为 name 的表单元素的值了。

例如，建立一个表单，设置 method 属性为 GET，添加一个文本框，命名为 user，获取表单元素的代码如下：

```
<?php
$user=$_GET["user"];                     //应用$_GET[]全局变量获取表单元素中文本框的值
?>
```

注意

PHP 可以应用$_POST[]或$_GET[]全局变量来获取表单元素的值。但值得注意的是，获取的表单元素名称区别字母大小写。如果读者在编写 Web 程序时疏忽了字母大小写，那么在程序运行时将获取不到表单元素的值或弹出错误提示信息。

8.4.3　$_SESSION[]变量

使用$_SESSION[]变量可以获取表单元素的值，格式为：

```
$_SESSION[name]
```

例如，建立一个表单，添加一个文本框，命名为 user，获取表单元素的代码如下：

```
$user=$_SESSION["user"]
```

使用$_SESSION[]传参的方法获取的变量值，保存之后任何页面都可以使用。但这种方法很耗费系统资源，建议读者慎重使用。关于$_SESSION 变量将在第 11 章进行详细讲解。

8.5 在 Web 页中嵌入 PHP 脚本

视频讲解：光盘\TM\lx\8\在 Web 页中嵌入 PHP 脚本.exe

在 Web 页中嵌入 PHP 脚本的方法有两种，一种是直接在 HTML 标记中添加<?php ?>PHP 标记符，写入 PHP 脚本，另一种是对表单元素的 value 属性进行赋值。

8.5.1 在 HTML 标记中添加 PHP 脚本

在 Web 编码过程中，可以随时添加 PHP 脚本标记<?php ?>，两个标记之间的所有文本都会被解释成为 PHP，而标记之外的任何文本都会被认为是普通的 HTML。

例如，在<body>标记中添加 PHP 标识符，使用 include 语句调用外部文件 top.php，代码如下：

```
<?php
include(" top.php ");                              //引用外部文件
?>
```

8.5.2 对表单元素的 value 属性进行赋值

在 Web 程序开发过程中，通常需要对表单元素的 value 属性进行赋值，以获取该表单元素的默认值。例如，为表单元素隐藏域进行赋值，只需要将所赋的值添加到 value 属性后即可，代码如下：

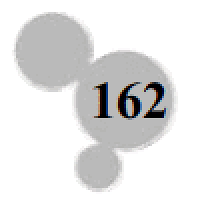
```
<?php
 $hidden="yg0025";                                 //为变量$hidden 赋值
?>
隐藏域的值 :<input type="hidden" name="ID" value="<?php echo $hidden;?>" >
```

从上面的代码中可以看出，首先为变量$hidden 赋予一个初始值，然后将变量$hidden 的值赋给隐藏域。在程序开发过程中，经常使用隐藏域存储一些无须显示的信息或需传送的参数。

8.6 在 PHP 中获取表单数据

视频讲解：光盘\TM\lx\8\在 PHP 中获取表单数据.exe

获取表单元素提交的值是表单应用中最基本的操作方法。本节中定义 POST 方法提交数据，对获



取表单元素提交的值进行详细讲解。

8.6.1　获取文本框、密码域、隐藏域、按钮、文本域的值

获取表单数据，实际上就是获取不同的表单元素的数据。<form>标签中的 name 是所有表单元素都具备的属性，即为这个表单元素的名称，在使用时需要使用 name 属性来获取相应的 value 属性值。所以，添加的所有控件必须定义对应的 name 属性值，另外，控件在命名上尽可能不要重复，以免获取的数据出错。

在程序开发过程中，获取文本框、密码域、隐藏域、按钮以及文本域的值的方法是相同的，都是使用 name 属性来获取相应的 value 属性值。本节仅以获取文本框中的数据信息为例，讲解获取表单数据的方法。希望读者能够举一反三，自行完成其他控件值的获取。

【例 8.5】 下面使用登录实例来学习如何获取文本框的信息。在下面的实例中，如果用户单击“登录”按钮，则获取用户名和密码。**（实例位置：光盘\TM\sl\8\5）**

具体开发步骤如下：

（1）利用开发工具（如 Dreamweaver）新建一个 PHP 动态页，并将其保存为 index.php。

（2）添加一个表单，添加一个文本框和一个提交按钮，代码如下：

```
<form name="form1" method="post" action="">
  用户名：<input type="text" name="user" size="20" ></td>
  密  码：<input name="pwd" type="password" id="pwd" size="20" > </td>
  <input name="submit" type="submit" id="submit" value="登录" > </td>
  </tr>
</table>
</form>
```

（3）在<form>表单元素外的任意位置添加 PHP 标记符，使用 if 条件语句判断用户是否提交了表单，如果条件成立，则使用 echo 语句输出使用$_POST[]方法获取的用户名和密码，代码如下：

```
<?php
if($_POST["submit"]=="登录"){                              //判断提交的按钮名称是否为“登录”
     //使用 echo 语句输出使用$_POST[]方法获取的用户名和密码
     echo "您输入的用户名为：".$_POST[user]."  密码为：".$_POST[pwd];
}
?>
```

注意

在应用文本框传值时，一定要正确地书写文本框的名称，其中不应该有空格；在获取文本框的提交值时，书写的文本框名称一定要与提交文本框页中设置的名称相同，否则将不能获取文本框的值。

（4）在 IE 浏览器中输入地址，按 Enter 键，运行结果如图 8.5 所示。



图 8.5　获取文本框、密码域、按钮的值

8.6.2　获取单选按钮的值

radio（单选按钮）一般是成组出现的，具有相同的 name 值和不同的 value 值，在一组单选按钮中，同一时间只能有一个被选中。

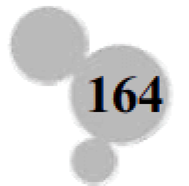
【例 8.6】 本实例中有两个 name=“sex”的单选按钮，选中其中一个并单击“提交”按钮，将会返回被选中的单选按钮的 value 值。**（实例位置：光盘\TM\sl\8\6）**

具体开发步骤如下：

（1）利用开发工具（如 Dreamweaver）新建一个 PHP 动态页，并将其保存为 index.php。

（2）添加一个表单，添加一组单选按钮和一个提交按钮，代码如下：

```
<form action="" method="post" name="form1">
        性别：
<input name="sex" type="radio" value="1" checked>男
         <input name="sex" type="radio" value="0">女
<input type="submit" name="Submit" value="提交">
</form>
```



说明

checked 属性是默认选中的意思。当表单页面被初始化时，有 checked 属性的表单元素为选中状态。

（3）在<form>表单元素外的任意位置添加 PHP 标记符，然后应用$_POST[]全局变量获取单选按钮组的值，最后通过 echo 语句进行输出，代码如下：

```
<?php
echo "您选择的性别为：";                    //输出字符串
echo $_POST["sex"];                         //输出被选中的单选按钮值
?>
```

（4）在 IE 浏览器中输入地址，按 Enter 键，运行结果如图 8.6 所示。



图 8.6　获取单选按钮的 value 值

8.6.3　获取复选框的值

复选框能够进行项目的多项选择。浏览者填写表单时，有时需要选择多个项目，例如，在线听歌中需要同时选取多个歌曲等，就会用到复选框。复选框一般都是多个同时存在，为了便于传值，name 的名字可以是一个数组形式，格式为：

```
<input type="checkbox" name="chkbox[]" value="chkbox1">
```

在返回页面可以使用 count()函数计算数组的大小，结合 for 循环语句输出选择的复选框的值。

【例 8.7】 本实例提供一组信息供用户选择，其中 name 值为 mrbook[]的数组变量。在处理页中显示出用户所选信息，如果数组为空，则返回“您没有选择”，实例代码如下：（**实例位置：光盘\TM\sl\8\7**）

具体开发步骤如下：

（1）新建一个 index.php 页面，创建 form 表单，添加一组复选框和一个提交按钮，代码如下：

```
<form name="form1" method="post" action="index.php">
<table width="445" cellpadding="0" cellspacing="0">
    <tr>
        <td width="443" height="41" align="center" valign="top">您喜欢的图书类型:
            <input type="checkbox" name="mrbook[]" value="入门类">入门类
            <input type="checkbox" name="mrbook[]" value="案例类">案例类
            <input type="submit" name="submit" value="提交">
        </td>
    </tr>
</table>
</form>
```

（2）在<form>表单元素外的任意位置添加 PHP 标记符，然后使用$_POST[]全局变量获取复选框的值，最后通过 echo 语句进行输出，代码如下：

```
<?php
if(($_POST[mrbook]!= null)){                              //判断复选框如果不为空，则执行下面操作
    echo "您选择的结果是：";                                //输出字符串
    for($i = 0;$i<count($_POST[mrbook]);$i++)            //通过 for 循环语句输出选中复选框的值
        echo $_POST[mrbook][$i]."  ";          //循环输出用户选择的图书类别
}
?>
```

（3）在 IE 浏览器中输入地址，按 Enter 键，运行结果如图 8.7 所示。

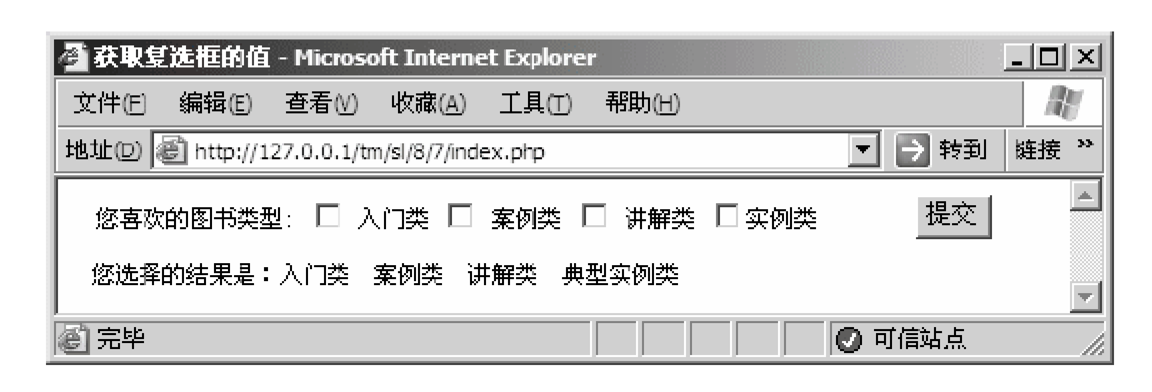
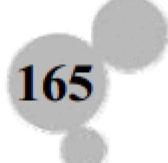


图 8.7　获取复选框的值



8.6.4 获取下拉列表框/菜单列表框的值

列表框有下拉列表框和菜单列表框两种形式，其基本的语法都一样。在进行网站程序设计时，下拉列表框和菜单列表框的应用非常广泛。可以通过下拉列表框和菜单列表框实现对条件的选择。

1．获取下拉列表框的值

获取下拉列表框的值的方法非常简单，与获取文本框的值类似，首先需要定义下拉列表框的 name 属性值，然后应用$_POST[]全局变量进行获取。

【例 8.8】 本实例是在下拉列表框中选择用户指定的条件，单击“提交”按钮，输出用户选择的条件值。**（实例位置：光盘\TM\sl\8\8）**

具体开发步骤如下：

（1）新建 index.php 页面，创建一个 form 表单，添加一个下拉列表框和一个提交按钮，实例代码如下：

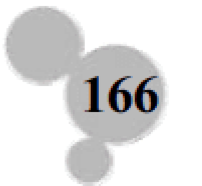
```
<form name="form1" method="post" action="">
<table width="280" border="0" cellpadding="0" cellspacing="0">
    <tr>
      <td width="80" height="20" align="center"><span class="style2">意见主题:</span></td>
      <td width="194">
          <select name="select" size="1">
                <option value="公司发展" selected>公司发展</option>
                <option value="管理制度">管理制度</option>
                <option value="后勤服务">后勤服务</option>
                <option value="员工薪资">员工薪资</option>
          </select>   
          <input type="submit" name="submit" value="提交">
          </td>
    </tr>
</table>
</form>
```

说明

在本例的代码中，在<select>标记中设置 size 属性，size 属性的值为 1，表示为下拉列表框；如果该值大于 1，则表示为列表框，以指定值的大小显示列表中的元素。如果列表中的元素大于 size 属性设置值，则自动添加垂直滚动条。

（2）编写 PHP 语句，通过$_POST[]全局变量获取下拉列表框的值，使用 echo 语句进行输出，代码如下：

```
<?php
if($_POST[select] != "")                        //响应表单事件，返回多选列表框的值
     echo "您选择的意见主题为：<br>";
```



```
    for($i = 0; $i < count($_POST[select]); $i++)
        echo $_POST[select][$i]."  ";
?>
```

（3）在 IE 浏览器中输入地址，按 Enter 键，运行结果如图 8.8 所示。



图 8.8　获取下拉列表框的值

2．获取菜单列表框的值

当<select>标记设置了 multiple 属性，则为菜单列表框，可以选择多个条件。由于菜单列表框一般都是多个值同时存在，为了便于传值，<select>标记的命名通常采用数组形式，格式为：

```
<input type="checkbox" name="chkbox[]" multiple>
```

在返回页面可以使用 count()函数计算数组的大小，结合 for 循环语句输出选择的菜单项。

【例 8.9】 本实例将设置一个菜单列表框，供用户选择喜欢的 PHP 类图书，单击“提交”按钮，输出选择的条件值。**（实例位置：光盘\TM\sl\8\9）**

具体开发步骤如下：

（1）新建一个 index.php 动态页，创建一个 form 表单，添加一个菜单列表框<select>，命名为“select []”的数组变量，添加一个提交按钮，实例代码如下：

```
<form name="form1" method="post" action="index.php">
  <table width="300" border="0" cellpadding="0" cellspacing="0">
    <tr>
      <td height="30" align="center" valign="middle">请选择您喜欢的 PHP 类图书</td>
    </tr>
    <tr>
      <td align="center" valign="middle"><select name="select[]" size="5" multiple>
          <option value="PHP 程序开发范例宝典">PHP 程序开发范例宝典</option>
          <option value="PHP 5 从入门到精通">PHP 5 从入门到精通</option>
          <option value="PHP 函数参考大全">PHP 函数参考大全</option>
        </select></td>
    </tr>
    <tr>
      <td height="30" align="center" valign="middle"><input type="submit" name="Submit" value="提交"></td>
    </tr>
  </table>
</form>
```

说明

在本例的代码中，在<select>标记中设置 multiple 属性，因此，size 属性的值与<option>标记的总数是对应的。

（2）编写 PHP 语句，通过$_POST[]全局变量获取菜单列表框的值，使用 echo 语句进行输出，代码如下：

```
<?php
if($_POST[submit]=="提交"){                              //使用 if 条件语句判断是否提交了表单
    echo "您选择的意见主题为：".$_POST[select];           //输出字符串和菜单列表框的值
}
?>
```

（3）在 IE 浏览器中输入地址，按 Enter 键，运行结果如图 8.9 所示。

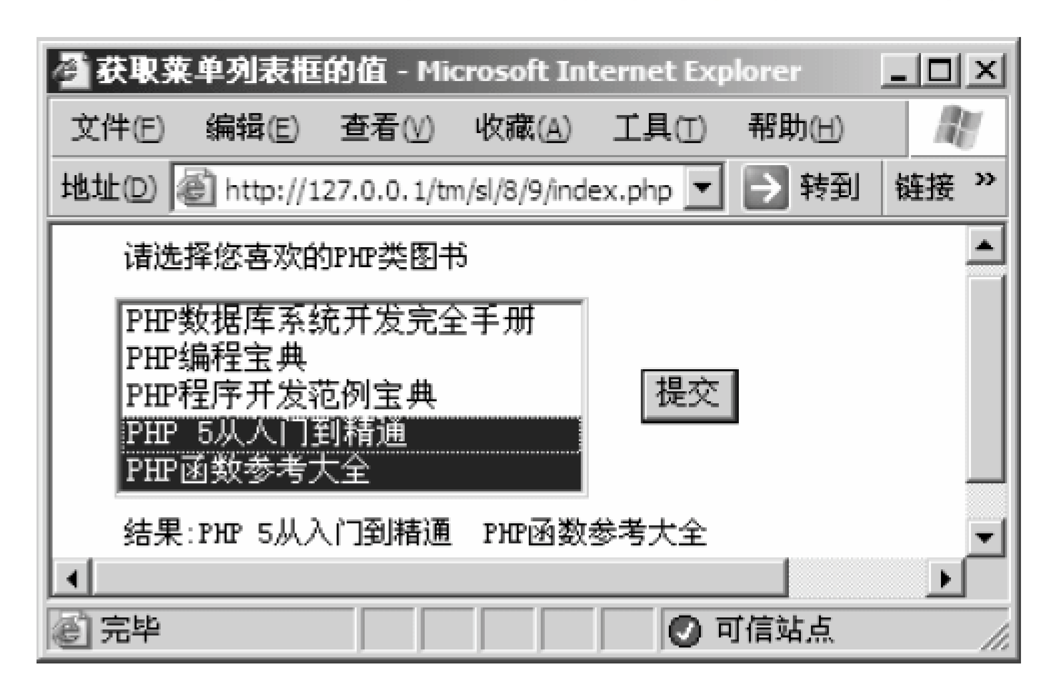


图 8.9　获取菜单列表框的值



技巧

读者可以按住 Shift 键或者是 Ctrl 键并单击，来选中多个菜单项。

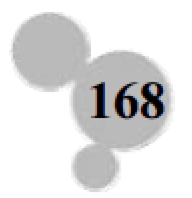
8.6.5　获取文件域的值

文件域的作用是实现文件或图片的上传。文件域有一个特有的属性 accept，用于指定上传的文件类型，如果需要限制上传文件的类型，则可以通过设置该属性完成。

【例 8.10】 在本实例中，选择需要上传的文件，单击“上传”按钮，就会在上方显示要上传文件的绝对路径。**（实例位置：光盘\TM\sl\8\10）**

具体开发步骤如下：

（1）新建 index.php 动态页，创建一个 form 表单，添加一个文件域和一个提交按钮，实例代码如下：



```
<form name="form1" method="post" action="index.php">
<input type="file" name="file" size="15" >
<input type="submit" name="upload" value="上传" >
</form>
```

说明

本实例实现的是获取文件域的值，并没有实现图片的上传，因此不需要设置<form>表单元素的enctype="multipart/form-data"属性。

（2）编写 PHP 代码，通过$_POST[]全局变量获取文件域的值，并通过 echo 语句输出上传文件的绝对路径，代码如下：



```
<?php
    echo $_POST[file];                                //输出要上传文件的绝对路径
?>
```

（3）在 IE 浏览器中输入地址，按 Enter 键，运行结果如图 8.10 所示。



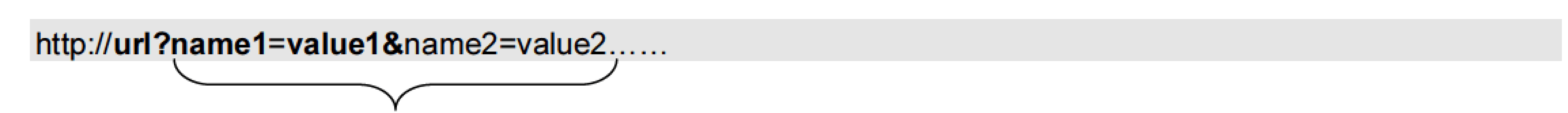
图 8.10　获取文件域的值

8.7　对 URL 传递的参数进行编/解码

视频讲解：光盘\TM\lx\8\对 URL 传递的参数进行编/解码.exe

8.7.1　对 URL 传递的参数进行编码

使用 URL 参数传递数据，就是在 URL 地址后面加上适当的参数。URL 实体对这些参数进行处理。使用方法如下：



显而易见，这种方法会将参数暴露，因此，本节针对该问题讲述一种 URL 编码方式，对 URL 传递的参数进行编码。

URL 编码是一种浏览器用来打包表单输入数据的格式，是对用地址栏传递参数进行的一种编码规则。如在参数中带有空格，则传递参数时就会发生错误，而用 URL 编码后，空格转换成%20，这样错误就不会发生了，对中文进行编码也是同样的情况，最主要的一点就是对传递的参数起到了隐藏的作用。

在 PHP 中对查询字符串进行 URL 编码，可以通过 urlencode()函数实现，该函数的语法如下：

```
string urlencode( string str)
```

urlencode()函数实现将字符串 str 进行 URL 编码。

【例 8.11】 本实例应用 urlencode()函数对 URL 传递的参数值"编程词典"进行编码，显示在 IE 地址栏中的字符串是 URL 编码后的字符串，实例代码如下：**（实例位置：光盘\TM\sl\8\11）**

```
<a href="index.php?id=<?php echo urlencode("编程词典");?>">PHP 编程词典</a>
```

运行结果如图 8.11 所示。

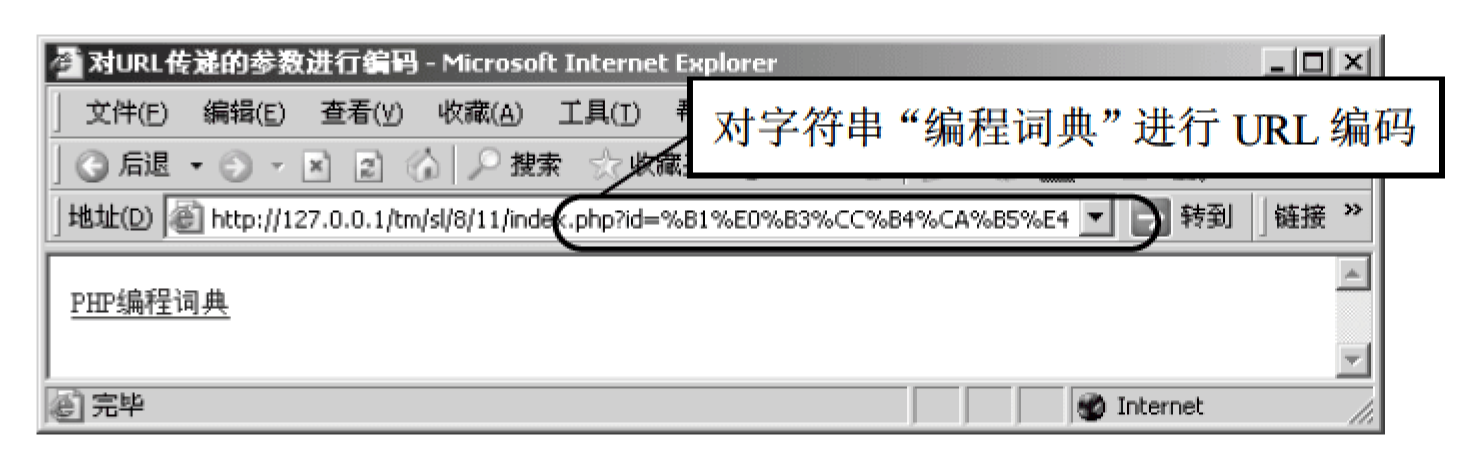


图 8.11　对 URL 传递的参数进行编码

说明

对于服务器而言，编码前后的字符串并没有什么区别，服务器能够自动识别。这里主要是为了讲解 URL 编码的使用方法。在实际应用中，对一些非保密性的参数不需要进行编码，读者可根据实际情况有选择地使用。

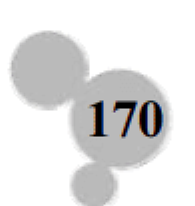
8.7.2　对 URL 传递的参数进行解码

对于 URL 传递的参数直接使用$_GET[]方法即可获取。而对于进行 URL 加密的查询字符串，则需要通过 urlencode()函数对获取后的字符串进行解码。该函数的语法如下：

```
string urlencode( string str)
```

urlencode()函数可将 URL 编码后的 str 查询字符串进行解码。

【例 8.12】 在例 8.11 中应用 urlencode()函数实现了对字符串"编码词典"进行编码，将编码后的字符串传给变量 id，本实例将应用 urldecode()函数对获取的变量 id 进行解码，将解码后的结果输出到浏览器，实例代码如下：**（实例位置：光盘\TM\sl\8\12）**



```
<a href="index.php?id=<?php echo urldecode("编程词典");?>">PHP 编程词典</a>
<?php echo "您提交的查询字符串的内容是:".urldecode($_GET[id]);?>
```

运行结果如图 8.12 所示。



图 8.12　对 URL 传递的参数进行解码

8.8　PHP 与 Web 表单的综合应用

视频讲解：光盘\TM\lx\8\PHP 与 Web 表单的综合应用.exe

表单是实现动态网页的一种主要的外在形式，使用表单可以收集客户端提交的信息，表单是网站互动功能的重要组成部分。

【例 8.13】 本实例将综合前面范例中介绍的有关表单中的各组件，实现各个组件的综合应用。本实例主要在例 8.2 的基础上，实现获取表单元素的值。通过 POST 方法将各个组件的值提交到本页，再通过$_POST 来获取提交的值。**（实例位置：光盘\TM\sl\8\13）**

具体开发步骤如下：

（1）表单的设计步骤可参见例 8.2 的开发步骤（1）～（3）。

（2）对表单提交的数据进行处理，输出各组件提交的数据，代码如下：

```
<?php
if($_POST[submit]!=""){                                  //如果提交了表单
     echo "您的个人简历内容是：";                          //输出字符串
     echo "  姓名:".$_POST[user];                         //输出用户名
     echo "  性别:".$_POST[sex];                          //输出性别
     echo "  密码:".$_POST[pwd];                          //输出密码
     echo "  学历:".$_POST[select];                       //输出学历
     echo "  爱好：";                                      //输出字符串
     //获取“爱好”复选框的值
     for($i=0;$i<count($_POST[fond]);$i++)
          echo $_POST[fond][$i]."  ";
         //实现文件上传功能，将上传的文件存储在 upfiles 文件夹中
         $path = './upfiles/'. $_FILES['photo']['name'];   //指定上传的路径及文件名
         move_uploaded_file($_FILES['photo']['tmp_name'],$path);  //上传文件
         echo "  个人写真：".$path;                        //输出个人写真的路径
          echo "  个人简介：".$_POST[intro];               //输出个人简历
     }
?>
```



关于上传图片或文件将在本书第 13 章进行详细讲解。

（3）在本实例的根目录下建立一个 upfiles 文件夹，用来存储上传的文件。

（4）在 IE 浏览器中输入地址，按 Enter 键，运行结果如图 8.13 所示。

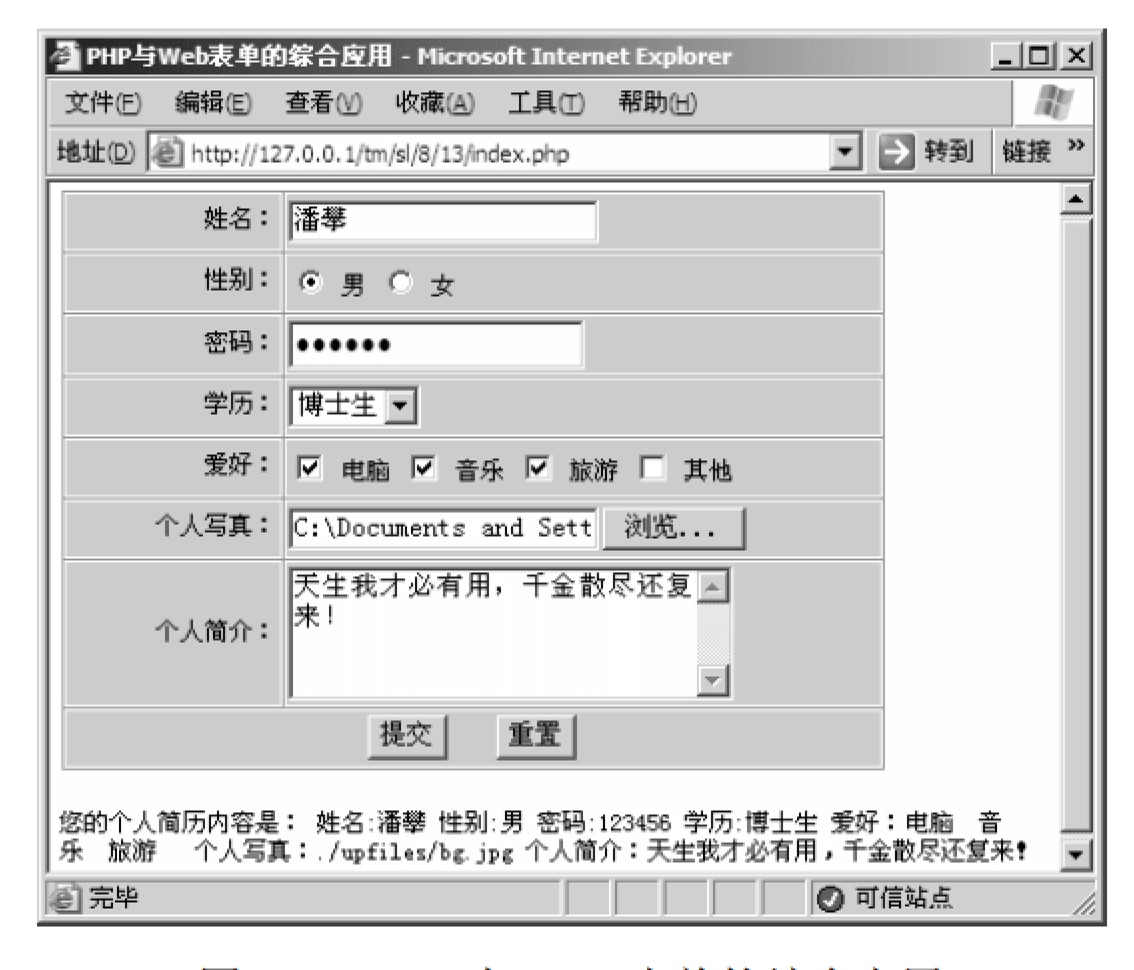


图 8.13　PHP 与 Web 表单的综合应用

8.9　小　　结

本章主要介绍了创建表单及表单元素、获取表单数据的方法、如何获取各种不同类型的表单数据，以及对 URL 传参的编码和解码。相信读者在学习完本章后，可以对表单应用自如，从而轻松实现“人机互交”。掌握了本章的技术要点，就意味着已经有了开发动态页的能力，为下一步的深入学习奠定良好的基础。

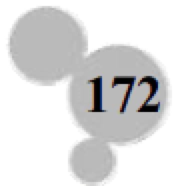
8.10　练习与实践

1．尝试创建一个表单，在表单中添加各个常用的元素，并为表单元素命名。**（答案位置：光盘\TM\sl\8\14）**

2．开发一个简单的搜索引擎页面，并获取输入的关键字。**（答案位置：光盘\TM\sl\8\15）**

3．开发一个页面，实现对 GET 方法传递的参数进行编码，然后对编码的字符串进行解码并输出。**（答案位置：光盘\TM\sl\8\16）**

4．开发一个用户注册页面，并输出用户的注册信息。**（答案位置：光盘\TM\sl\8\17）**



第 9 章

PHP 与 JavaScript 交互

（视频讲解：1 小时 10 分钟）

JavaScript 是一种可以嵌入在 HTML 代码中由客户端浏览器运行的脚本语言。在网页中使用 JavaScript 代码，不仅可以实现网页特效，还可以响应用户请求实现动态交互的功能。在 PHP 动态网页中灵活运用 JavaScript，可以实现更强大的功能。本章将介绍 JavaScript 脚本语言的基础知识，使读者在掌握基础内容的前提下能够熟练运用 JavaScript 制作 Web 页面。

通过阅读本章，您可以：

- 了解 JavaScript 是什么
- 了解 JavaScript 能做什么
- 了解 JavaScript 脚本语言的基础
- 灵活运用 JavaScript 实现自定义函数
- 熟练使用 JavaScript 条件控制语句
- 熟练使用 JavaScript 循环控制语句
- 熟悉调用 JavaScript 跳转语句
- 掌握在网页中执行 JavaScript、调用自定义函数和引用 JS 文件的方法
- 了解解决浏览器不支持 JavaScript 语言的方法
- 熟练运用在 PHP 动态页中调用 JavaScript 脚本

9.1 了解 JavaScript

视频讲解：光盘\TM\lx\9\了解 JavaScript.exe

JavaScript 是脚本编程语言，支持 Web 应用程序的客户端和服务器端构件的开发，在 Web 系统中得到了非常广泛的应用。下面对 JavaScript 进行简单的介绍。

9.1.1 什么是 JavaScript

JavaScript 是由 Netscape Communication Corporation（网景公司）开发的，是一种基于对象和事件驱动并具有安全性能的解释型脚本语言。它不但可用于编写客户端的脚本程序，由 Web 浏览器解释执行，而且还可以编写在服务器端执行的脚本程序，在服务器端处理用户提交的信息并动态地向浏览器返回处理结果。

9.1.2 JavaScript 的功能

JavaScript 是比较流行的一种制作网页特效的脚本语言，它由客户端浏览器解释执行，可以应用在 PHP、ASP、JSP 和 ASP.NET 网站中，同时目前比较热门的 Ajax 就是以 JavaScript 为基础，由此可见，熟练掌握并应用 JavaScript 对于网站开发人员非常重要。

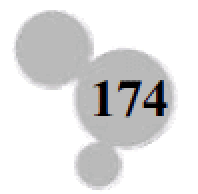
JavaScript 主要应用于以下几个方面：

☑ 在网页中加入 JavaScript 脚本代码，可以使网页具有动态交互的功能，便于网站与用户间的沟通，及时响应用户的操作，对提交的表单做即时检查，如验证表单元素是否为空，验证表单元素是否是数值型、检测表单元素是否输入错误等。
☑ 应用 JavaScript 脚本制作网页特效，如动态的菜单、浮动的广告等，为页面增添绚丽的动态效果，使网页内容更加丰富、活泼。
☑ 应用 JavaScript 脚本建立复杂的网页内容，如打开新窗口载入网页。
☑ 应用 JavaScript 脚本可以对用户的不同事件产生不同的响应。
☑ 应用 JavaScript 制作各种各样的图片、文字、鼠标、动画和页面的效果。
☑ 应用 JavaScript 制作一些小游戏。

9.2 JavaScript 语言基础

视频讲解：光盘\TM\lx\9\JavaScript 语言基础.exe

JavaScript 脚本语言与其他语言一样，有其自身的基本数据类型、表达式和运算符以及程序的基本



框架结构。通过本节的学习可以使读者掌握更多的 JavaScript 脚本语言的基础知识。

9.2.1　JavaScript 数据类型

JavaScript 主要有 6 种数据类型，如表 9.1 所示。

表 9.1　JavaScript 数据类型

数 据 类 型	说　　明	举　　例
字符串型	使用单引号或双引号括起来的一个或多个字符	如"PHP"、"I like study PHP"等
数值型	包括整数或浮点数（包含小数点的数或科学记数法的数）	如-128、12.9、6.98e6 等
布尔型	布尔型常量只有两种状态，即 true 或 false	如 event.returnValue=false
对象型	用于指定 JavaScript 程序中用到的对象	如网页表单元素
Null 值	可以通过给一个变量赋 null 值来清除变量的内容	如 a=null
Undefined	表示该变量尚未被赋值	如 var a

9.2.2　JavaScript 变量

变量是指程序中一个已经命名的存储单元，它的主要作用就是为数据操作提供存放信息的容器。在使用变量前，必须明确变量的命名规则、变量的声明方法及变量的作用域。

1．变量的命名规则

JavaScript 变量的命名规则如下：

☑　必须以字母或下划线开头，中间可以是数字、字母或下划线。
☑　变量名不能包含空格或加号、减号等符号。
☑　JavaScript 的变量名是严格区分大小写的。例如，User 与 user 代表两个不同的变量。
☑　不能使用 JavaScript 中的关键字。JavaScript 的关键字如表 9.2 所示。

表 9.2　JavaScript 的关键字

abstract	continue	finally	instanceof	private	this
boolean	default	float	int	public	throw
break	do	for	interface	return	typeof
byte	double	function	long	short	true
case	else	goto	native	static	var
catch	extends	implements	new	super	void
char	false	import	null	switch	while
class	final	in	package	synchronized	with



虽然 JavaScript 的变量可以任意命名，但为了在编程时使代码更加规范，最好使用便于记忆且有意义的变量名称，以增加程序的可读性。

2. 变量的声明与赋值

在 JavaScript 中，一般使用变量前需要先声明变量，但有时变量可以不必先声明，在使用时根据变量的实际作用来确定其所属的数据类型。所有的 JavaScript 变量都由关键字 var 声明。

语法如下：

```
var variable;
```

在声明变量的同时也可以对变量进行赋值：

```
var variable=11;
```



建议读者在使用变量前就对其声明，因为声明变量的最大好处就是能及时发现代码中的错误。由于 JavaScript 是采用动态编译的，而动态编译是不易于发现代码中的错误的，特别是变量命名方面的错误。

声明变量时所遵循的规则如下：

可以使用一个关键字 var 同时声明多个变量，例如：

```
var i,j;
```

可以在声明变量的同时对其赋值，即为初始化，例如：

```
var i=1;j=100;
```

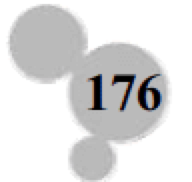
如果只是声明了变量，并未对其赋值，则其值默认为 undefined。

注意

在 JavaScript 中，可以使用分号代表一个语句的结束，如果每个语句都在不同的行中，那么分号可以省略；如果多个语句在同一行中，那么分号就不能省略。建议读者不省略分号，以养成良好的编程习惯。

如声明 3 个不同数据类型的变量，代码如下：

```
var i=100;                                  //定义变量 i 为数值类型
var str="有一条路，走过了总会想起";             //定义变量 str 为字符串类型
var content=true;                           //定义变量 content 为布尔类型
```



注意

在程序开发过程中，可以使用 var 语句多次声明同一个变量，如果重复声明的变量已经有一个初始值，那么此时的声明变量就相当于对变量重新赋值。

9.2.3　JavaScript 注释

视频讲解：光盘\TM\lx\9\JavaScript 注释.exe

在 JavaScript 中，采用的注释方法有两种：

1．单行注释

单行注释使用“//”进行标识。“//”符号后面的文字都不被程序解释执行。例如：

```
//这里是程序代码的注释
```

2．多行注释

多行注释使用“/*…*/”进行标识。“/*…*/”符号后面的文字不被程序解释执行。例如：

```
/*
这里是多行程序注释
*/
```

注意

多行注释“/*…*/”中可以嵌套单行注释“//”，但不能嵌套多行注释“/*…*/”。因为第一个“/*”会与其后面第一个“*/”相匹配，从而使后面的注释不起作用，甚至引起程序出错。

另外，JavaScript 还能识别 HTML 注释的开始部分“<!--”，JavaScript 会将其看作单行注释结束，如使用“//”一样。但 JavaScript 不能识别 HTML 注释的结尾部分“-->”。

这种现象存在的主要原因是：在 JavaScript 中，如果第一行以“<!--”开始，最后一行以“-->”结束，那么其间的程序就包含在一个完整的 HTML 注释中，会被不支持 JavaScript 的浏览器忽略掉，不能被显示。如果第一行以“<!--”开始，最后一行以“//-->”结束，JavaScript 会将两行都忽略掉，而不会忽略这两行之间的部分。用这种方式可以针对那些无法理解 JavaScript 的浏览器而隐藏代码，而对那些可以理解 JavaScript 的浏览器则不必隐藏。

9.3　自定义函数

视频讲解：光盘\TM\lx\9\自定义函数.exe

自定义函数就是由用户自己命名并编写的能实现特定功能的程序单元。用户使用的自定义函数必

须事先声明，不能直接使用没有声明过的自定义函数。

JavaScript 用 function 来定义函数，语法格式如下：

```
function 函数名([参数]){    函
    return var;             数
}                           体
```

自定义函数的调用方法是：

```
函数名();
```

其中的括号一定不能省略。

【例 9.1】 本实例将自定义一个 calculate()函数，实现两个数的乘积，然后在函数体外调用自定义函数 calculate()，向自定义函数中传递两个参数，最后应用 document.write()对象输出结果，实例代码如下：**（实例位置：光盘\TM\sl\9\1）**

```
<script language="javascript">
function calculate(a,b){                    //自定义一个 calculate()函数
    return a*b;                             //返回两个参数的乘积
}
document.write(calculate(15,15));           //调用 calculate()函数并传递参数，输出结果
</script>
```

结果为：225



在同一个页面中不能定义名称相同的函数。另外，当用户自定义函数后，需要对该函数进行引用，否则自定义的函数将失去意义。

9.4 JavaScript 流程控制语句

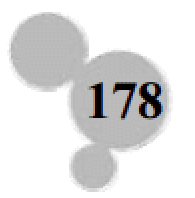
视频讲解：光盘\TM\lx\9\JavaScript 流程控制语句.exe

流程控制语句就是对语句中不同条件的值进行判断，从而根据不同的条件执行不同的语句。在 JavaScript 中，流程控制语句可以分为条件语句、循环语句和跳转语句。

9.4.1 条件语句

条件控制语句主要包括两种：一种是 if 条件语句，另一种是 switch 多分支语句。

在 JavaScript 中，可以使用单一的 if 条件语句，也可以使用两个或者多重选择的 if 条件语句。



1．if 条件语句

if 语句是最基本、最常用的条件控制语句。通过判断条件表达式的值为 true 或者 false，来确定是否执行某一条语句。

语法格式如下：

```
语    if(条件表达式){
句  {  语句块
块    }
```

在 if 语句中，只有当条件表达式的值为 true 时，才会执行“语句块”中的语句，否则将跳过语句块，执行其他程序语句。其中，大括号“{}”的作用是将多条语句组成一个语句块，作为一个整体进行处理。如果语句块中只有一条语句，也可以省略大括号。一般情况下，建议不要省略大括号，以免出现程序错误。

例如，首先定义一个变量，并且设置变量的值为空，然后使用 if 语句判断变量的值，如果值等于空，则弹出提示信息“变量的内容为空”，否则没有任何信息输出。代码如下：

```
var form="";
if(form==""){
	alert("变量的内容为空！");
}
```

运行结果：变量的内容为空！

下面通过具体的实例讲解在页面中嵌入 JavaScript 脚本代码，从而及时响应用户的操作。

【例 9.2】 创建一个表单元素，添加一个下拉列表框，命名为 year，在<input>标记的属性中添加 onclick 事件，调用自定义函数 check()，在该函数中使用 if 条件语句判断指定的年份是否为闰年，实例代码如下：**（实例位置：光盘\TM\sl\9\2）**

```
<form name="form1" method="post" action="">
      <span class="style2">检测闰年：</span>
      <select name="year">
      <option value="2000">2000 年</option>
…                                                    <!-- 省略了部分表单元素代码 -->
      <option value="2008" selected>2008 年</option>
   </select>
   <input type="submit" name="Submit" value="检测" onclick="check();">
</form>
```

在<body>标记外，添加 JavaScript 脚本自定义的函数 check()，在 if 语句中通过给出的表达式判断变量 year 所代表的年份是否为闰年，即如果变量值能够被 4 整除并且不能被 100 整除，则说明为闰年。代码如下：

```
<script language="javascript">
function check(){
var year1=form1.year.value;                    //定义变量 year1，并获取提交的表单元素的值
	//如果变量 year1 能够被 4 整除，而同时不能被 100 整除，则执行下面的语句
	if((year1%4)==0 && (year1%100)!=0){
```

```
            alert(year1+"年是闰年！");                    //如果 year1 变量满足条件，则输出此年份为闰年
        }
}
</script>
```

在 IE 浏览器中输入地址，按 Enter 键，在下拉列表框中选择“2008 年”，单击“检测”按钮，运行结果如图 9.1 所示。



图 9.1　应用 if 条件语句判断指定的年份是否为闰年

技巧

如果大括号中只有一条语句，那么大括号{}可以被省略。建议不要省略大括号，养成使用大括号的习惯，可以避免一些无意中造成的错误。

除了上面讲解的标准的 if 单一条件语句外，if…else 语句也是 if 语句的标准形式，是双分支条件语句。语法格式如下：

```
if(条件表达式)
条件为真 {
        语句块 1;
    }
else
条件为假 {
        语句块 2;
    }
```

在 if…else 语句中，当条件表达式的值为 true 时，将执行“语句块 1”中的语句；当“条件表达式”的值为 false 时，将跳过“语句块 1”而执行“语句块 2”中的语句。

在例 9.2 中，可以使用 if…else 语句对提交的年份进行判断，如果是闰年则弹出某年是闰年的提示；如果不是闰年，则弹出某年是平年的提示。更改后的代码如下：

```
<script language="javascript">
function check(){
    var year1=form1.year.value;                //定义变量 year1，并获取提交的表单元素的值
    //如果变量 year1 能够被 4 整除，而同时不能被 100 整除，则执行下面的语句
    if((year1%4)==0 && (year1%100)!=0){
        alert(year1+"年是闰年！");              //如果 year1 变量满足条件，则输出此年份为闰年
    }else{
        alert(year1+"年是平年!");               //如果 year1 变量不满足条件，则输出此年份为平年
    }
```

```
}
</script>
```

代码中加粗的部分为 if 条件语句的分支部分。

2．switch 分支语句

虽然使用 if 语句可以实现多分支的条件语句，但在选择分支比较多的情况下，使用 if 多分支条件语句就会降低程序的执行效率。JavaScript 中的 switch 语句可以针对给出的表达式或者变量的不同值来选择执行的语句块，从而提高程序运行速度。

语法格式如下：

```
switch(表达式或变量){
  case  常量表达式 1:
语           语句块 1;
句           break;
块
  case 常量表达式 2:
          语句块 2;
          break;
     …
case 常量表达式 n:
          语句块 n;
          break;
     default:
          语句块 n+1;
          break;
}
```

在 switch 语句中，首先计算表达式或变量的值，然后将此值与“常量表达式 1”进行比较，如果两个值相等，则执行“语句块 1”中的语句，然后执行 break 语句并跳出 switch 语句；如果此值与“常量表达式 1”不相等，则将此值与“常量表达式 2”进行比较，如果相等，则执行“语句块 2”中的语句，并执行 break 语句跳出 switch 语句；如果与“常量表达式 2”不相等，则继续与后面的常量表达式进行比较。如果表达式或变量的值与所有 case 语句后的“常量表达式”都不相等，则执行 default 中的“语句块 n+1”。

【例 9.3】本实例将创建一个表单，添加一组单选按钮，命名为 book，在<input>标记中添加 onclick 事件，调用自定义函数 check()，并将单选按钮的值传到自定义函数中，实例代码如下：（**实例位置：光盘\TM\sl\9\3**）

```
<form name="form1" method="post" action="">
     <span class="style2">您最喜爱的图书类别：</span>
     <input name="book" type="radio" value="生活类" onclick="check(this.value);">生活类
     <input name="book" type="radio"value="电脑类" onclick="check(this.value);">电脑类
     <input name="book" type="radio" value="科技类" onclick="check(this.value);">科技类
     <input name="book" type="radio"value="体育类" onclick="check(this.value);">体育类
</form>
```

在<body>标记外，添加 JavaScript 脚本自定义的函数 check()，应用 switch 语句判断变量的值与 case 标签的值是否匹配，如果对比的值匹配，则输出 case 标签后的内容，代码如下：

```
<script language="javascript">
function check(books){
	switch(books){
		case "生活类":
			alert("您最喜爱的图书类别是:"+books);
		break;
		case "电脑类":
			alert("您最喜爱的图书类别是:"+books);
		break;
		case "科技类":
			alert("您最喜爱的图书类别是:"+books);
		break;
		case "体育类":
			alert("您最喜爱的图书类别是:"+books);
		break;
	}
}
</script>
```

在 IE 浏览器中输入地址，按 Enter 键，选中“电脑类”单选按钮，即可弹出用户选择的结果，如图 9.2 所示。



图 9.2　应用 switch 语句输出选择条件

9.4.2　循环语句

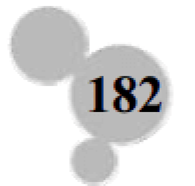
循环语句主要功能是在满足条件的情况下反复地执行某一个操作。循环语句主要包括 while 循环语句和 for 循环语句。

1．while 循环语句

while 语句是基本的循环语句，也是条件判断语句。在 JavaScript 中，while 循环语句的应用比较广泛。语法格式如下：

```
语   while(条件表达式){
句       语句块
块   }
```

在 while 语句中，首先判断条件表达式的值，如果值为 true 则执行大括号内的语句块，执行完毕后再次判断条件表达式的值，如果值仍为 true，则重复执行大括号内的语句块。这样一直循环，直到条件表达式的值为 false 时结束，执行 while 语句后面的其他代码。



注意

在 while 语句的循环体中应包含改变条件表达式值的语句，否则条件表达式的值总为 true，会造成死循环。

【例 9.4】 应用 while 循环语句输出变量 i 的值，实例代码如下：（**实例位置：光盘\TM\sl\9\4**）

```
<script language="javascript" type="text/javascript">
    var i=3;                                //定义变量 i，并赋初始值
    while (i>0){                            //定义 while 语句中的逻辑表达式为 i>0
        document.write("-"+i);              //调用 document 对象的 write 方法输出变量 i 的值
        i--;                                //执行 i--运算，变量 i 的值逐次减 1
    }
</script>
```

结果为：-3-2-1

2．for 循环语句

for 语句是一种常用的循环控制语句。在 for 语句中，可以应用循环变量来明确循环的次数和具体的循环条件。for 语句通常使用一个变量作为计数器来执行循环的次数，这个变量就称为循环变量。

语法格式如下：

```
语    for (初始化循环变量;循环条件;确定循环变量的改变值){
句        语句块;
块    }
```

在 for 语句的小括号中包含 3 部分内容：

- ☑ 初始化循环变量：该表达式的作用是声明循环变量并进行初始化赋值。在 for 语句之前也可以对循环变量进行声明和赋值。
- ☑ 循环条件：该表达式是基于循环变量的一个条件表达式，如果条件表达式的返回值为 true，则执行循环体内的语句块。循环体内的语句执行完毕后将重新判断此表达式，直到条件表达式的返回值为 false 时终止循环。
- ☑ 确定循环变量的改变值：该条件表达式用于操作循环变量的改变值。每次执行完循环体内的语句后，在判断循环条件之前，都将执行此表达式。

注意

for 语句可以使用 break 语句来终止循环语句的执行。break 语句默认情况下是终止当前的循环语句。

【例 9.5】 使用 for 循环语句输出变量 i 叠加相乘的表达式及结果值，实例代码如下：（**实例位置：光盘\TM\sl\9\5**）

```
<script language="javascript">
for(i=1;i<=9;i++){                          //初始化变量 i，定义循环条件，变量 i 递增
```

```
        document.write(i+"*"+i+ "="+i*i+"  ");      //输出变量 i 叠加相乘的表达式及结果
}
</script>
```

在上面代码中，在 for 语句中定义了变量 i 和变量 i 的初始值；定义循环条件为 i<=9，即在 i<=9 的情况下执行循环体中的语句；定义变量 i 的值为每循环一次累加 1。在循环体中，通过调用 document 对象的 write 方法输出变量 i 叠加相乘的表达式与结果。

在 IE 浏览器中输入地址，按 Enter 键，运行结果如图 9.3 所示。

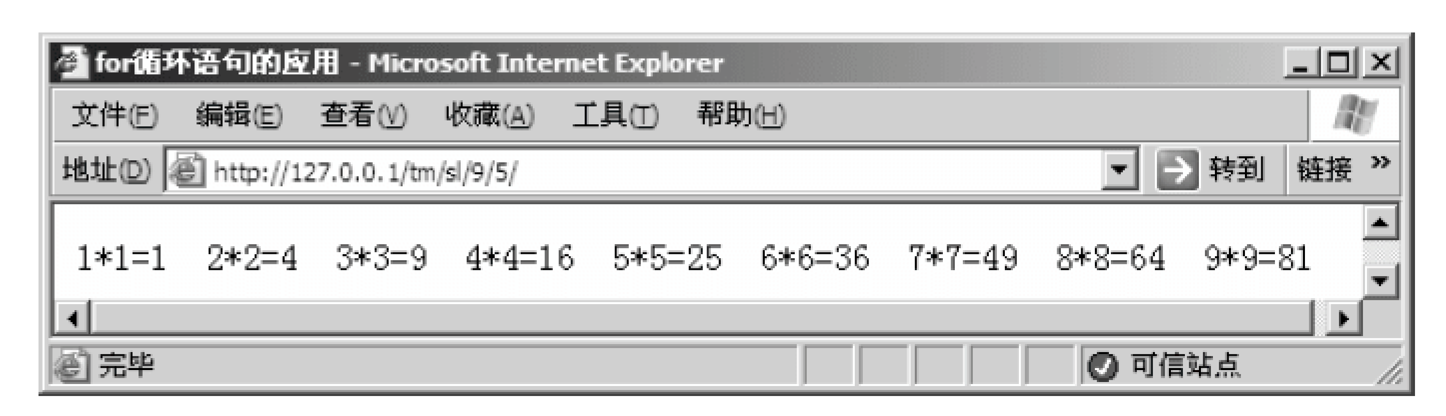


图 9.3　for 循环语句的应用

9.4.3　跳转语句

跳转语句是在循环控制语句的循环体中的指定位置或是满足一定条件的情况下直接退出循环。JavaScript 跳转语句分为 break 语句和 continue 语句。

1．break 语句

break 语句用来终止执行其后面的程序并跳出循环，或者结束 switch 语句。

语法格式如下：

```
break;
```

【例 9.6】 在 for 循环语句中，当循环变量 i 的值大于 10 时退出 for 循环，实例代码如下：**（实例位置：光盘\TM\sl\9\6）**

```
<script language="javascript" >
for( i=0;i<20;i++ ){            //在 for 语句中初始化循环变量，定义循环条件，定义每次循环后变量 i 的值累加 1
        if(i>10){
                break;                                  //如果 i>10 就会立即跳出循环
        }
        document.write(i+"-");                          //输出 i 的值
}
</script>
```

在上面的代码中，当变量 i 的值大于 10 时调用 break 语句，这时程序将跳出 for 循环不再执行下面的循环。如果未使用 break 语句，程序将执行 for 循环语句中的循环体，直到变量 i 的值不满足条件 i<20。

注意

在嵌套的循环语句中使用 break 语句时，break 语句只能跳出最近的一层循环，而不是跳出所有的循环。

结果为：0-1-2-3-4-5-6-7-8-9-10-

2．continue 语句

continue 语句与 break 语句的作用不同。continue 语句是只跳出本次循环并立即进入到下一次循环；break 语句则是跳出循环后结束整个循环。

语法格式如下：

```
continue;
```

【例 9.7】 输出指定范围内的奇数，实例代码如下：（**实例位置：光盘\TM\sl\9\7**）

```
<script language="javascript">
    var str="20 以内的偶数有：";          //定义变量 str
    var i=1;                            //定义变量 i
    while(i<20){                        //应用 while 语句，定义循环条件为 i<10
        if(i%2!=0){                     //如果变量 i 能被 2 整除，则执行下面的语句
            i++;                        //在退出本次循环之前使变量 i 的值累加 1，默认该语句将导致死循环
            continue;                   //调用 continue 语句
        }
            str=str+i+" ";              //拼接字符串 str，以获取变量 i 的值
            i++;                        //使变量 i 的值累加 1
    }
    document.write(str);                //输出变量 str 的值
</script>
```

在上面代码中，首先初始化变量 i；然后在 while 循环语句中先使用 if 语句判断变量 i 是否能被 2 整除，如果不能被 2 整除（说明此值为奇数）则使变量 i 的值累加 1，并调用 continue 语句跳出本次循环进入到下一个循环，如果变量 i 能被 2 整除（说明此值为偶数）则获取变量 i 的值，并使变量 i 的值累加 1；当变量 i 的值不满足条件 i<20 时将结束 while 循环。

结果为：20 以内的偶数有：2 4 6 8 10 12 14 16 18

9.5　JavaScript 事件

视频讲解：光盘\TM\lx\9\JavaScript 注释.exe

JavaScript 是基于对象的语言。它的一个最基本的特征就是采用事件驱动。事件是某些动作发生时产生的信号，这些事件随时都可能发生。引起事件发生的动作称之为触发事件，例如，当鼠标指针经过某个按钮、用户单击了某个链接、用户选中了某个复选框、用户在文本框中输入某些信息等，都会触发相应的事件。

为了便于读者查找 JavaScript 中的常用事件，下面以表格的形式对各事件进行说明，如表 9.3 所示。

表 9.3　JavaScript 中的常用事件

状　　态	事　　件	说　　明
鼠标键盘事件	onclick	鼠标单击时触发此事件
	ondblclick	鼠标双击时触发此事件
	onmousedown	按下鼠标时触发此事件
	onmouseup	鼠标按下后松开鼠标时触发此事件
	onmouseover	当鼠标移动到某对象范围的上方时触发此事件
	onmousemove	鼠标移动时触发此事件
	onmouseout	当鼠标离开某对象范围时触发此事件
	onkeypress	当键盘上的某个按键被按下并且释放时触发此事件
	onkeydown	当键盘上的某个按键被按下时触发此事件
	onkeyup	当键盘上的某个按键被按下后松开时触发此事件
页面相关事件	onabort	图片在下载时被用户中断触发此事件
	onload	页面内容完成时触发此事件（也就是页面加载事件）
	onresize	当浏览器的窗口大小被改变时触发此事件
	onunload	当前页面将被改变时触发此事件
表单相关事件	onblur	当前元素失去焦点时触发此事件
	onchange	当前元素失去焦点并且元素的内容发生改变时触发此事件
	onfocus	当某个元素获得焦点时触发此事件
	onreset	当表单中 reset 的属性被激活时触发此事件
	onsubmit	一个表单被提交时触发此事件
滚动字幕事件	onbounce	当 Marquee 内的内容移动至 Marquee 显示范围之外时触发此事件
	onfinish	当 Marquee 元素完成需要显示的内容后触发此事件
	onstart	当 Marquee 元素开始显示内容时触发此事件



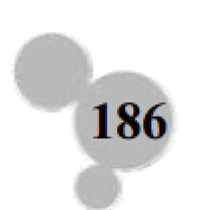
在 PHP 中应用 JavaScript 脚本中的事件调用自定义函数是程序开发过程中经常使用的方法。

9.6　调用 JavaScript 脚本（JavaScript 脚本嵌入方式）

视频讲解：光盘\TM\lx\9\调用 JavaScript 脚本.exe

9.6.1　在 HTML 中嵌入 JavaScript 脚本

JavaScript 作为一种脚本语言，可以使用<script>标记嵌入到 HTML 文件中。



语法格式如下：

```
<script language="javascript">
…
</script>
```

应用<script>标记是直接执行 JavaScript 脚本最常用的方法，大部分含有 JavaScript 的网页都采用这种方法，其中，通过 language 属性可以设置脚本语言的名称和版本。

注意

如果在<script>标记中未设置 language 属性，Internet Explorer 浏览器和 Netscape 浏览器将默认使用 JavaScript 脚本语言。

【例 9.8】 本实例将实现在 HTML 中嵌入 JavaScript 脚本，这里直接在<script>和</script>标记中间写入 JavaScript 代码，用于弹出一个提示对话框，实例代码如下：**（实例位置：光盘\TM\sl\9\8）**

```
<html>
<head>
<title>在 HTML 中嵌入 JavaScript 脚本</title>
</head>
<body>
<script language="javascript">
	alert("我很想学习 PHP 编程，请问如何才能学好这门语言!");
</script>
</body>
</html>
```

在上面的代码中，<script>与</script>标记之间调用 JavaScript 脚本语言 window 对象的 alert 方法，向客户端浏览器弹出一个提示对话框。这里需要注意的是，JavaScript 脚本通常写在<head>…</head>标记和<body>…</body>标记之间。写在<head>标记中间的一般是函数和事件处理函数；写在<body>标记中间的是网页内容或调用函数的程序块。

在 IE 浏览器中打开 HTML 文件，运行结果如图 9.4 所示。



图 9.4　在 HTML 中嵌入 JavaScript 脚本

在 HTML 中通过“javascript:”可以调用 JavaScript 的方法。例如，在页面中插入一个按钮，在该按钮的 onclick 事件中应用“javascript:”调用 window 对象的 alert 方法，弹出一个警告提示框，代码如下：

```
<input type="submit" name="Submit" value="单击这里" onClick="javascript:alert('您单击了这个按钮！ ')">
```

9.6.2　应用 JavaScript 事件调用自定义函数

在 Web 程序开发过程中，经常需要在表单元素相应的事件下调用自定义函数。例如，在按钮的单击事件下调用自定义函数 check()来验证表单元素是否为空，代码如下：

```
<input type="submit" name="Submit" value="检测" onClick="check();">
```

然后在该表单的当前页中编写一个 check()自定义函数即可。自定义函数在 9.3 节已经详细介绍过，这里不再赘述。另外，关于 JavaScript 的常用事件请参见本章 9.5 节。

9.6.3　在 PHP 动态网页中引用 JS 文件

在网页中，除了可在<script>与</script>标记之间编写 JavaScript 脚本代码，还可以通过<script>标记中的 src 属性指定外部的 JavaScript 文件（即 JS 文件，以.js 为扩展名）的路径，从而引用对应的 JS 文件。

语法格式如下：

```
<script src＝url   language="Javascript"></script>
```

其中，url 是 JS 文件的路径，“language="Javascript"”可以省略，因为<script>标记默认使用的就是 JavaScript 脚本语言。

JavaScript 脚本不仅可以与 HTML 结合使用，同时也可以与 PHP 动态网页结合使用，其引用的方法是相同的。使用外部 JS 文件的优点如下：

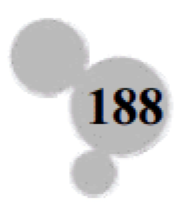
☑ 使用 JS 文件可以将 JavaScript 脚本代码从网页中独立出来，便于代码的阅读。

☑ 一个外部 JS 文件，可以同时被多个页面调用。当共用的 JavaScript 脚本代码需要修改时，只需要修改 JS 文件中的代码即可，便于代码的维护。

☑ 通过<script>标记中的 src 属性不但可以调用同一个服务器上的 JS 文件，还可以通过指定路径来调用其他服务器上的 JS 文件。

【例 9.9】 本实例将在网页中通过<script>标记的 src 属性引用外部 JS 文件，用于弹出一个提示对话框。index.php 文件中的代码如下：**（实例位置：光盘\TM\sl\9\9）**

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>在 PHP 动态网页中引用 JS 文件</title>
</head>
<script src="script.js"></script>
<body>
</body>
</html>
```



在同级目录下创建一个 script.js 文件，代码如下：

```
alert("恭喜您，成功调用了 script.js 外部文件!");
```

从上面的代码可以看出，在 index.php 文件中通过设定<script>标记中的 src 属性，引用了同级目录下的 script.js 文件。在 script.js 文件中调用 JavaScript 脚本语言 window 对象的 alert 方法，在客户端浏览器弹出一个提示对话框。

在 IE 浏览器中输入地址，按 Enter 键，运行结果如图 9.5 所示。



图 9.5　在 PHP 动态网页中引用 JS 文件

在网页中引用 JS 文件需要注意的事项如下：

☑ 在 JS 文件中，只能包含 JavaScript 脚本代码，不能包含<script>标记和 HTML 代码。读者可参考例 9.9 中 script.js 文件的代码。

☑ 在引用 JS 文件的<script>与</script>标记之间不应存在其他的 JavaScript 代码，即使存在，浏览器也会忽略此脚本代码，而只执行 JS 文件中的 JavaScript 脚本代码。

9.6.4　解决浏览器不支持 JavaScript 的问题

虽然大多数浏览器都支持 JavaScript 脚本，但仍有少部分浏览器不支持。如果遇到不支持 JavaScript 脚本的浏览器，网页会达不到预期效果或出现错误。解决这个问题可以使用以下 3 种方法。

1．开启 IE 浏览器对 JavaScript 的支持

目前有些支持 JavaScript 的浏览器出于安全考虑关闭了对 JavaScript 的支持。这时，浏览者可以启用对 JavaScript 脚本的支持来解决这一问题。具体操作步骤如下：

（1）启动 IE 浏览器，选择“工具”/“Internet 选项”命令，打开“Internet 选项”对话框，选择“安全”选项卡，选择 Internet 安全设置项，单击“自定义级别”按钮，打开如图 9.6 所示的对话框。

（2）将对话框中的“Java 小程序脚本”和“活动脚本”两个选项设置为启用状态。单击“确定”按钮，即可开启 IE 浏览器支持 JavaScript 脚本的功能。

2．开启 IE 浏览器对本地 JavaScript 的支持

IE 浏览器将网页分为 Internet、本地 Intranet、受信任的站点和受限制的站点 4 个区域，但不包括本地网页。通常在 Windows XP 操作系统中，在 IE 浏览器中打开包含 JavaScript 脚本的网页时，会弹出如图 9.7 所示的安全提示对话框。

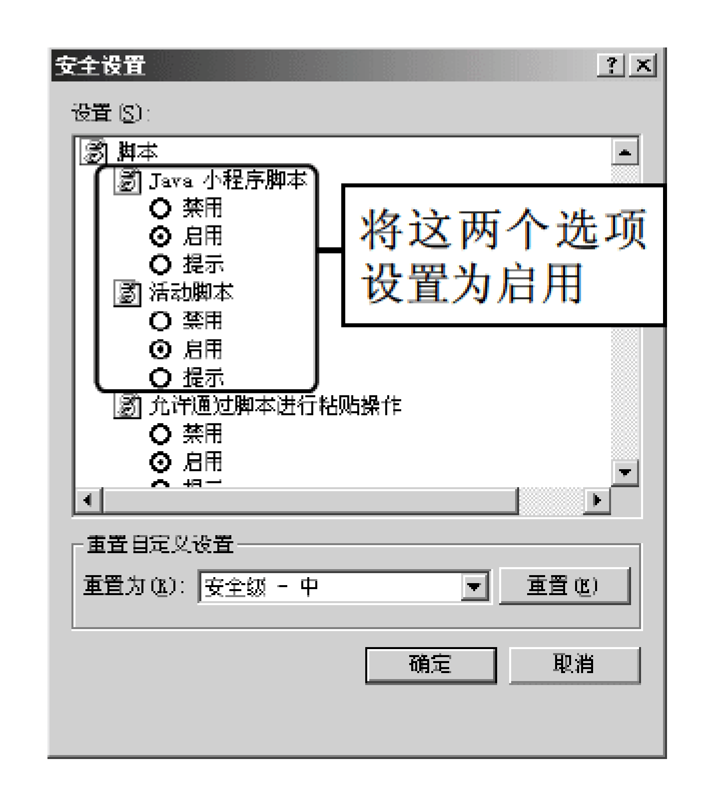



图 9.6　启用 JavaScript 脚本功能

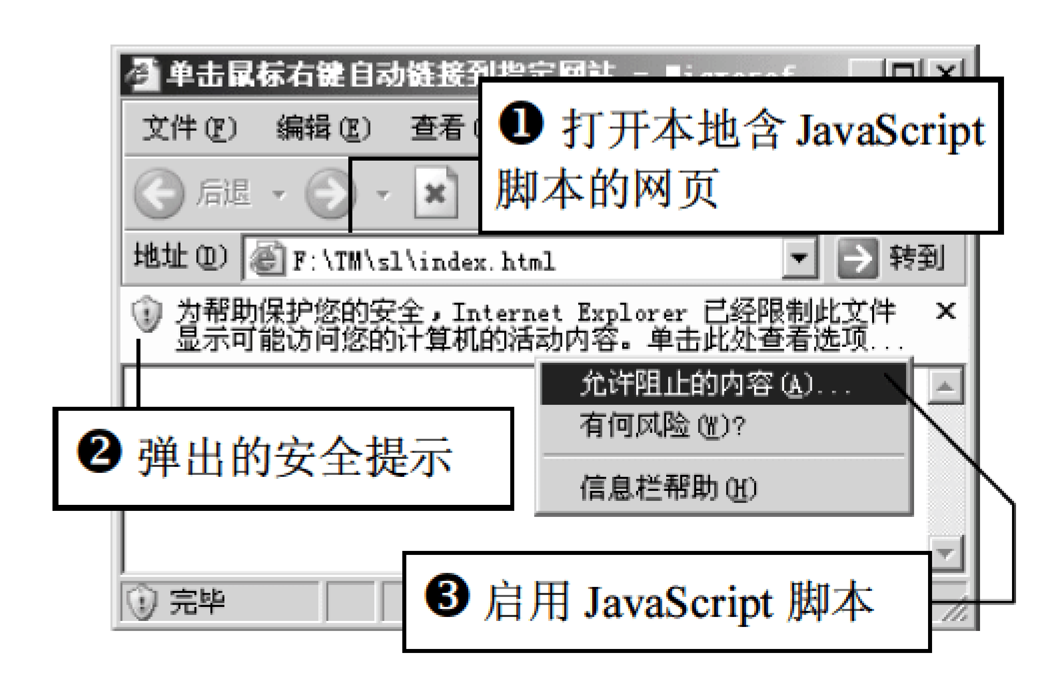


图 9.7　安全提示对话框

如果用户要继续执行网页中包含的 JavaScript 脚本，可以右击安全提示区域，在弹出的快捷菜单中选择“允许阻止的内容”命令，如图 9.7 所示，即可成功运行本网站。但此选项仅针对当前网页，若要永久地消除 IE 浏览器的这种安全提示，需要对 IE 浏览器做如下设置：

在 IE 浏览器中选择“工具”/“Internet 选项”命令，打开“Internet 选项”对话框。选择“高级”选项卡，在安全设置区选中“允许活动内容在我的计算机上的文件中运行”和“允许来自 CD 的活动内容在我的计算机上运行”复选框（此选项仅适用于 Windows XP 操作系统），单击“确定”按钮，即可成功解决上述问题。

3．应用注释符号验证浏览器是否支持 JavaScript 脚本功能

如果用户不能确定自己的浏览器是否支持 JavaScript 脚本，那么可以使用 HTML 提供的注释符号进行验证。HTML 注释符号是以“<!--”开始、以“-->”结束的。如果在此注释符号内编写 JavaScript 脚本，对于不支持 JavaScript 的浏览器，将会把编写的 JavaScript 脚本作为注释处理。

【例 9.10】 使用 JavaScript 脚本在页面中输出一个字符串，将 JavaScript 脚本编写在 HTML 注释中，如果浏览器支持 JavaScript 则输出此字符串；如果不支持则不输出此字符串。实例代码如下：**（实例位置：光盘\TM\sl\9\10）**

```
<html>
<head>
<title>应用注释符号验证浏览器是否支持 JavaScript 脚本功能</title>
</head>
<body>
<script type="text/javascript" >
<!--
	document.write("您的浏览器支持 JavaScript 脚本！");
-->
</script>
</body>
</html>
```

在 IE 浏览器中输入地址，按 Enter 键，运行结果如图 9.8 所示。

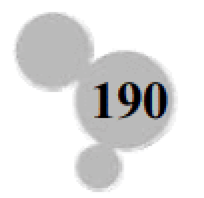




图 9.8　应用注释符号验证浏览器是否支持 JavaScript 脚本

4．应用<noscript>标记验证浏览器是否支持 JavaScript 脚本

如果用户不能确定浏览器是否支持 JavaScript 脚本，可以使用<noscript>标记进行验证。

如果当前浏览器支持 JavaScript 脚本，那么该浏览器将会忽略<noscript>…</noscript>标记之间的任何内容。如果浏览器不支持 JavaScript 脚本，那么浏览器将会把<noscript>…</noscript>标记之间的内容显示出来。通过此标记可以提醒浏览者当前使用的浏览器是否支持 JavaScript 脚本。

【例 9.11】 使用 JavaScript 脚本在页面中输出一个字符串，并使用<noscript>标记提醒浏览者当前浏览器是否支持 JavaScript 脚本。实例代码如下：**（实例位置：光盘\TM\sl\9\11）**

```
<html>
<head>
<title>应用&lt;noscript&gt;标记验证浏览器是否支持 JavaScript 脚本</title>
</head>
<body>
<script language="javascript">
	document.write("您的浏览器支持 JavaScript 脚本");
</script>
<noscript>
	您的浏览器不支持 JavaScript 脚本
</noscript>
</body>
</html>
```

在 IE 浏览器中输入地址，按 Enter 键，运行结果如图 9.9 所示。



图 9.9　应用<noscript>标记验证浏览器是否支持 JavaScript 脚本

技巧

当解释程序遇到</script>标记时会终止当前脚本。要显示"</script>" 本身，可将“<”改写为“<”，将“>”改写为“>”。若要使用 document.write 输出<script>…</script>标记，需要将闭合标记通过反斜杠进行转义，如<script>…<\/script>。

9.7　在 PHP 中调用 JavaScript 脚本

视频讲解：光盘\TM\lx\9\在 PHP 中调用 JavaScript 脚本.exe

9.7.1　应用 JavaScript 脚本验证表单元素是否为空

在程序开发过程中，经常要应用 JavaScript 脚本来判断表单提交的数据是否为空，或者判断提交的数据是否符合标准等。

【例 9.12】 本实例主要通过 if 语句和 form 对象的相关属性验证表单元素是否为空。（**实例位置：光盘\TM\sl\9\12**）

具体开发步骤如下：

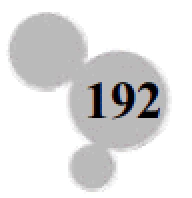
（1）设计表单页，添加一个表格并设置表格的背景图片路径为 images/bg.jpg，添加一个用户名文本框并命名为 user，添加一个密码域并命名为 pwd，代码如下：

```
<form name="myform" method="post" action="">
  <table width="532" height="183" align="center" cellpadding="0" cellspacing="0" bgcolor="#CCFF66" background=
"images/bg.jpg">
     <tr><td height="71" colspan="2" align="center"> </td></tr>
     <tr>
         <td width="281" align="left">
            用户名：<input name="user" type="text" id="user" size="20"> <br><br>
            密  码：<input name="pwd" type="password" id="pwd" size="20">
         </td>
     </tr>
     <tr>
        <td height="43" align="center">
            <input type="submit" name="submit" onClick="return mycheck();" value="登录"> 
            <input type="reset" name="Submit2" value="重置">
         </td>
     </tr>
  </table>
</form>
```

（2）在上面的代码中，在“登录”按钮的表单元素中添加了一个 onclick 鼠标单击事件，调用自定义函数 mycheck()，代码如下：

```
<input type="submit" name="submit" onClick="return mycheck();" value="登录">
```

（3）在<form>表单元素外应用 function 定义一个函数 mycheck()，用来验证表单元素是否为空。在 mycheck()函数中，应用 if 条件语句判断表单提交的用户名和密码是否为空，如果为空，则弹出提示信息，自定义函数如下：



```
<script language="javascript">
function mycheck(){                                                        //定义一个函数
      if(myform.user.value==""){                                           //通过 if 语句判断用户名是否为空
      alert("用户名称不能为空！！");myform.user.focus();return false;      //返回表单元素位置
      }
      if(myform.pwd.value==""){                                            //通过 if 语句判断密码是否为空
            alert("用户密码不能为空！！");myform.pwd.focus();return false;}  //返回表单元素位置
}
</script>
```

（4）在 IE 浏览器中输入地址，按 Enter 键，单击“登录”按钮，运行结果如图 9.10 所示。



图 9.10　应用 JavaScript 脚本验证表单元素是否为空



说明

本实例中介绍的只是通过 JavaScript 脚本验证表单元素是否为空，还可以通过 JavaScript 脚本验证表单元素值的格式是否正确，例如验证电话号码的格式、邮箱地址的格式等，类似的实例可以参考第 6 章 6.5 节的内容。

9.7.2　应用 JavaScript 脚本制作二级导航菜单

应用 JavaScript 脚本不仅可以用来验证表单元素，而且可以制作各式各样的网站导航菜单。本节以网站开发中最常用的二级导航菜单为例，讲解其实现方法。

【例 9.13】 本实例主要应用 JavaScript 的 switch 语句确定要显示的二级菜单的内容。（**实例位置：光盘\TM\sl\9\13**）

具体开发步骤如下：

（1）在网页中适当的位置添加一级导航菜单，本例中的一级导航菜单是由一系列空的超链接组成，这些空的超链接执行的操作是调用自定义的 JavaScript 函数 Lmenu()显示对应的二级菜单，在调用时需要传递一个标记，即主菜单项的参数，代码如下：

```
<table width="761" height="20" border="0" cellpadding="0" cellspacing="0">
  <tr>
       <td width="67" align="center"><a href="index.php">首 页</a></td>
       <td width="75" align="center"><a href="#" onMouseMove="Lmenu('新品')">新品上架</a></td>
       <td width="75" align="center"><a href="#" onMouseMove="Lmenu('购物')">购物车</a></td>
       <td width="74" align="center"><a href="#" onMouseMove="Lmenu('会员')">会员中心</a></td>
       <td width="61" align="center"><a href="index.php">在线帮助</a></td>
   </tr>
 </table>
```

（2）在网页中要显示二级菜单的位置添加一个名为 submenu 的 div 层，代码如下：

```
<div id="submenu" class="word_yellow"> </div>
```

（3）编写自定义的 JavaScript 函数 Lmenu()，用于在鼠标移动到某个一级菜单时，根据传递的参数值在页面中指定的位置显示对应的二级菜单，并设置二级菜单的名称及链接文件，代码如下：

```
<script language="javascript">
function Lmenu(value){
  switch (value){
     case "新品":
       submenu.innerHTML="<a href='#'>商品展示</a>|<a href='#'>销售排行榜</a>|<a href='#'>商品查询</a>";
       break;
     case "购物":
       submenu.innerHTML=" <a href='#'>添加商品</a>|<a href='#'>移出指定商品</a>|<a href='#'>清空购物车
</a>|<a href='#'>查询购物车</a>|<a href='#'>填写订单信息</a> ";
        break;
     case "会员":
       submenu.innerHTML="<a href='#'>注册会员</a>|<a href='#'>修改会员</a>|<a href='#'>账户查询</a>";
       break;
   }
}
</script>
```

在自定义函数 Lmenu()中，首先计算 switch 语句括号内表达式的值，当此表达式的值与某个 case 后面的常数表达式的值相等时，就执行此 case 后的语句，从而实现二级菜单。当执行某个 case 后的语句时，如果遇到 break 语句，则结束这条 switch 语句的执行，转去执行这条 switch 语句之后的语句。


注意

通常情况下，应该在 switch 语句的每个分支后面都加上 break，使 JavaScript 只执行匹配的分支。

（4）在 IE 浏览器中输入地址并按 Enter 键，当鼠标指针移动到一级菜单“购物车”超链接上时，在页面的指定位置显示“添加商品”、“移出指定商品”、“清空购物车”、“查询购物车”、“填写订单信息”等购物车的二级子菜单，运行结果如图 9.11 所示。

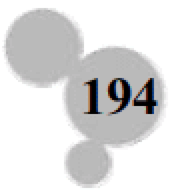




图 9.11　应用 JavaScript 脚本制作二级导航菜单

9.7.3　应用 JavaScript 脚本控制文本域和复选框

在动态网站的开发过程中，经常需要对文本域中的内容进行清空或者修改，选中多个复选框进行提交等操作。这里介绍一种通过 JavaScript 脚本控制文本域内容和复选框勾选的方法。

【例 9.14】 在本实例中，通过 JavaScript 脚本实现清空文本域中的值、复选框的全选、反选和不选。**（实例位置：光盘\TM\sl\9\14）**

具体开发步骤如下：

（1）创建一个 form 表单，添加文本域和多个复选框。在文本域后添加一个超链接，应用 onClick 事件调用 JavaScript 脚本中 document.getElementById 标记，为文本域赋值为空，清空文本域；添加图像域，通过 onClick 事件调用不同的方法，实现复选框的全选、反选和不选。

```
<form method="post" name="form1" id="form1" action="">
<table width="547" border="1" cellpadding="1" cellspacing="1" bordercolor="#FFFFFF" bgcolor="#FBA720">
    <tr>
        <td height="35" colspan="5" bgcolor="#FFFFFF"><span class="STYLE1">订单管理</span></td>
    </tr>
    <tr>
        <td width="77" align="right" bgcolor="#FFFFFF">说明：</td>
        <td width="389" ><textarea name="readme" cols="50" rows="10" id="readme"></textarea></td>
        <td width="63" height="33" bgcolor="#FFFFFF" class="STYLE2">
            <a href="#" onClick="javascript:document.getElementById('readme').value='';return false;">
<img src="images/_14.jpg" width="60" height="25" border="0" /></a>
        </td>
    </tr>
    <tr>
        <td rowspan="6" align="right" bgcolor="#FFFFFF">操作：</td>
        <td height="30" colspan="2" align="left" bgcolor="#FFFFFF"><input name="PHP3" type="checkbox"
id="PHP3" value="PHP" />C++编程词典全能版
        </td>
    </tr>
```

```
		<tr>
			<td colspan="5" align="center" bgcolor="#FFFFFF">
				<img src="images/_01.jpg" onclick="checkAll(form1,status)" width="60" height="25" />
				<img src="images/_08.jpg" onclick="switchAll(form1,status)" width="60" height="25" />
				<img src="images/_11.jpg" width="60" height="25" onclick="uncheckAll(form1,status)" />
			</td>
		</tr>
</table>
</form>
```

（2）编写 JavaScript 脚本，定义 3 个方法：checkAll()、switchAll()和 uncheckAll()，用于实现复选框的全选、反选和不选。代码如下：

```
<script language="javascript">
function checkAll(form1,status){							//全选
	var elements = form1.getElementsByTagName('input');		//获取 input 标签
	for(var i=0; i<elements.length; i++){					//根据标签的长度执行循环
		if(elements[i].type == 'checkbox'){					//判断对象中元素的类型，如果类型为 checkbox
			if(elements[i].checked==false){					//判断当 checked 的值为 false 时
				elements[i].checked=true;					//为 checked 赋值为 true
			}
		}
	}
}
function switchAll(form1,status){							//反选
	var elements = form1.getElementsByTagName('input');
	for(var i=0; i<elements.length; i++){
		if(elements[i].type == 'checkbox'){
			if(elements[i].checked==true){
				elements[i].checked=false;
			}else if(elements[i].checked==false){
				elements[i].checked=true;
			}
		}
	}
}
function uncheckAll(form1,status){							//不选
	var elements = form1.getElementsByTagName('input');		//获取 input 标签
	for(var i=0; i<elements.length; i++){					//根据标签的长度执行循环
		if(elements[i].type == 'checkbox'){					//判断对象中元素的类型，如果类型为 checkbox
			if(elements[i].checked==true){					/判断当 checked 的值为 true 时
				elements[i].checked=false;					//为 checked 赋值为 false
			}
		}
	}
}
</script>
```

运行结果如图 9.12 所示。

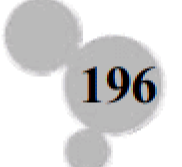




图 9.12　应用 JavaScript 脚本控制文本域和复选框

9.8　小　　结

通过本章的学习，读者可以了解到 JavaScript 是什么、能做什么以及 JavaScript 语言的基础。本章重点介绍了在 HTML 静态页和 PHP 动态页中调用 JavaScript 脚本的不同方法，以及如何自定义函数和灵活运用 JavaScript 流程控制语句。在熟悉和掌握了各个知识点后，相信读者能够举一反三，在 PHP 与 JavaScript 脚本语言的交互下开发更实用的网络程序。

9.9　练习与实践

1．创建一个 PHP 动态页面，添加以“博客”为主题的各表单元素，当用户单击“发表”按钮时，调用自定义函数 check()，验证各表单元素是否为空。（**答案位置：光盘\TM\sl\9\15**）

2．在 PHP 动态页中引用 JS 文件来动态显示系统的当前时间。（**答案位置：光盘\TM\sl\9\16**）

3．应用 JavaScript 脚本控制输入字符串的长度。（**答案位置：光盘\TM\sl\9\17**）

第10章

日期和时间

（视频讲解：25 分钟）

在 Web 开发中对日期和时间的使用与处理是必不可少的。例如，在电子商务网站上查看最新商品、在论坛中查看最新主题、定时删除 Session 等。这些都是和时间密不可分的。在世界上各个地区对时间的表示也不尽相同，如英语中的 Sunday，汉语表示为星期日，韩语则为일요일等，如果都要手动来处理是不现实的，在 PHP 中提供了本地化日期时间的概念。

通过阅读本章，您可以：

- 掌握系统的时区设置
- 掌握如何获取本地时间戳
- 掌握如何获取当前日期和时间
- 掌握如何获取日期信息
- 掌握如何将日期和时间解析为 UNIX 时间戳
- 熟悉比较两个时间的大小
- 熟悉倒计时功能
- 了解如何计算页面运行时间

10.1 系统时区设置

视频讲解：光盘\TM\lx\10\系统时区设置.exe

10.1.1 时区划分

整个地球分为 24 个时区，每个时区都有自己的本地时间。同一时间，每个时区的本地时间相差 1～23 小时，例如，英国伦敦本地时间与北京本地时间相差 8 个小时。在国际无线电通信领域，使用一个统一的时间，称为通用协调时间（UTC，Universal Time Coordinated），UTC 与格林威治标准时间（GMT，Greenwich Mean Time）相同，都与英国伦敦的本地时间相同。

10.1.2 时区设置

由于 PHP 5.0 对 data()函数进行了重写，因此，目前的日期时间函数比系统时间少 8 个小时。在 PHP 语言中默认设置的是标准的格林威治时间（即采用的是零时区），所以要获取本地当前的时间必须更改 PHP 语言中的时区设置。

更改 PHP 语言中的时区设置有两种方法：

（1）修改 php.ini 文件中的设置，找到[date]下的“;date.timezone =”选项，将其修改为“date.timezone =Asia/Hong_Kong”，然后重新启动 Apache 服务器。

（2）在应用程序中，在使用时间日期函数之前添加如下函数：

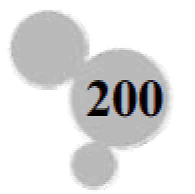
```
date_default_timezone_set(timezone);
```

参数 timezone 为 PHP 可识别的时区名称，如果时区名称 PHP 无法识别，则系统采用 UTC 时区。在 PHP 手册中提供了各时区名称列表，其中，设置我国北京时间可以使用的时区包括：PRC（中华人民共和国）、Asia/Chongqing（重庆）、Asia/Shanghai（上海）或者 Asia/Urumqi（乌鲁木齐），这几个时区名称是等效的。

设置完成后，date()函数便可以正常使用，不会再出现时差问题。

注意

如果将程序上传到空间中，那么对系统时区设置时，不能修改 php.ini 文件，只能使用 date_default_timezone_set()函数对时区进行设置。



10.2　PHP 日期和时间函数

视频讲解：光盘\TM\lx\10\PHP 日期和时间函数.exe

PHP 提供了大量的内置函数，使开发人员在日期和时间的处理上游刃有余，大大提高了工作效率。在本节中，将介绍一些常用的 PHP 日期和时间函数及实际应用的实例。

10.2.1　获得本地化时间戳

PHP 应用 mktime()函数将一个时间转换成 UNIX 的时间戳值。

mktime()函数根据给出的参数返回 UNIX 时间戳。时间戳是一个长整数，包含了从 UNIX 纪元（1970 年 1 月 1 日）到给定时间的秒数。其参数可以从右向左省略，任何省略的参数会被设置成本地日期和时间的当前值。该函数的语法格式如下：

```
int mktime(int hour, int minute, int second, int month, int day, int year, int [is_dst] )
```

mktime()函数的参数说明如表 10.1 所示。

表 10.1　mktime()函数的参数说明

参　数	说　明
hour	小时数
minute	分钟数
second	秒数（一分钟之内）
month	月份数
day	天数
year	年份数，可以是两位或 4 位数字，0～69 对应于 2000～2069，70～100 对应于 1970～2000
is_dst	参数 is_dst 在夏令时可以被设置为 1，如果不是则设置为 0；如果不确定是否为夏令时则设置为 -1（默认值）

注意

有效的时间戳典型范围是格林威治时间 1901 年 12 月 13 日 20:45:54～2038 年 1 月 19 日 03:14:07（此范围符合 32 位有符号整数的最小值和最大值）。在 Windows 系统中此范围限制为从 1970 年 1 月 1 日～2038 年 1 月 19 日。

【例 10.1】 本例使用 mktime()函数获取系统的当前时间，由于返回的是时间戳，还要通过 date()函数对其进行格式化，才能够输出日期和时间，实例代码如下：**（实例位置：光盘\TM\sl\10\1）**

```
<?php
    echo "mktime 函数返回的时间戳：".mktime()."<p>";                    //返回当前的时间戳
```

```
    echo "当前的日期为：".date("Y-m-d",mktime())."<p>";          //使用 date 函数输出格式化后的日期
    echo "当前的时间为：".date("H:i:s",mktime());                //使用 date 函数输出格式化后的时间
?>
```

运行结果如图 10.1 所示。



图 10.1　使用 mktime()函数获取当前时间

10.2.2　获取当前时间戳

PHP 通过 time()函数获取当前的 UNIX 时间戳，返回值为从 UNIX 纪元（格林威治时间 1970 年 1 月 1 日 00:00:00）到当前时间的秒数。

语法格式如下：

```
int time ( void )
```

【例 10.2】 本例中使用 time()函数获取当地时间戳，并将时间戳格式化输出。实例代码如下：**（实例位置：光盘\TM\sl\10\2）**

```
<?php
$nextWeek = time() + (7 * 24 * 60 * 60);             //7 days; 24 hours; 60 mins; 60secs
echo 'Now：date('Y-m-d') ."<p>";                      //输出当前日期
echo 'Next Week: '. date('Y-m-d', $nextWeek);         //输出变量 NextWeek 的日期
?>
```

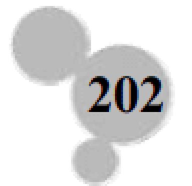
运行结果如图 10.2 所示。



图 10.2　使用 time()函数获取当前时间戳

10.2.3　获取当前日期和时间

在 PHP 中通过 date()函数获取当前的日期和时间。date()函数的语法如下：



```
date(string format,int timestamp)
```

date()函数将返回参数 timestamp 按照指定格式而产生的字符串。其中的参数 timestamp 是可选的，如果省略，则使用当前时间。format 参数可以使开发人员按其指定的格式输出日期时间，关于 format 参数的格式化选项将在 10.2.6 节进行介绍，这里给出几个预定义常量，如表 10.2 所示，这几个常量提供了标准的日期表达方法，可用于日期格式函数。

表 10.2　关于时间日期的预定义常量

函　　数	说　　明
DATE_ATOM	原子钟格式
DATE_COOKIE	HTTP Cookies 格式
DATE_ISO8601	ISO-8601 格式
DATE_RFC822	RFC822 格式
DATE_RFC850	RFC850 格式
DATE_RSS	RSS 格式
DATE_W3C	World Wide Web Consortium 格式

【例 10.3】本实例将比较各个常量的输出有什么区别。实例代码如下：（**实例位置：光盘\TM\sl\ 10\3**）

```
<?php
    echo "DATE_ATOM = ".date(DATE_ATOM);                        //输出 ATOM 格式的日期
    echo "<p>DATE_COOKIE = ".date(DATE_COOKIE);                 //输出 HTTP Cookie 格式的日期
    echo "<p>DATE_ISO8601 = ".date(DATE_ISO8601);               //输出 ISO8601 格式的日期
    echo "<p>DATE_RFC822 = ".date(DATE_RFC822);                 //输出 RFC822 格式的日期
    echo "<p>DATE_RFC850 = ".date(DATE_RFC850);                 //输出 RFC850 格式的日期
    echo "<p>DATE_RSS = ".date(DATE_RSS);                       //输出 RSS 格式的日期
    echo "<p>DATE_W3C = ".date(DATE_W3C);                       //输出 W3C 格式的日期
?>
```

运行结果如图 10.3 所示。



图 10.3　预定义常量

注意

也许有的读者得到的时间和系统时间并不相同，这是因为在 PHP 语言中默认设置的是标准的格林威治时间，而不是北京时间。如果出现了时间不符的情况，可参考 10.1 节的系统时区设置。

10.2.4　获取日期信息

日期是数据处理中经常使用到的信息之一。本节主要应用 getdate()函数获取日期指定部分的相关信息。getdate()函数的语法如下：

```
array getdate(int timestamp)
```

getdate()函数返回数组形式的日期和时间信息，如果没有时间戳，则以当前时间为准。该函数返回的关联数组元素的说明如表 10.3 所示。

表 10.3　getdate()函数返回的关联数组元素说明

函　　数	说　　明
seconds	秒，返回值为 0～59
minutes	分钟，返回值为 0～59
hours	小时，返回值为 0～23
mday	月份中第几天，返回值为 1～31
wday	星期中第几天，返回值为 0（表示星期日）～6（表示星期六）
mon	月份数字，返回值为 1～12
year	4 位数字表示的完整年份，返回的值如 2000 或 2008
yday	一年中第几天，返回值为 0～365
weekday	星期几的完整文本表示，返回值为 Sunday～Saturday
month	月份的完整文本表示，返回值为 January～December
0	返回从 UNIX 纪元开始的秒数

【例 10.4】 下面使用 getdate()函数获取系统当前的日期信息，并输出该函数的返回值，实例代码如下：**（实例位置：光盘\TM\sl\10\4）**

```
<?php
	$arr = getdate();                                                    //使用 getdate()函数将当前信息保存
	echo $arr[year]."-".$arr[mon]."-".$arr[mday]." ";                    //返回当前的日期信息
	echo $arr[hours].":".$arr[minutes].":".$arr[seconds]." ".$arr[weekday];//返回当前的时间信息
	echo "<p>";
	echo "Today is the $arr[yday]th of year";                            //输出今天是一年中的第几天
?>
```

运行结果如图 10.4 所示。





图 10.4　getdate()函数获取时间日期信息

10.2.5　检验日期的有效性

一年有 12 个月、一个月有 31 天（或 30 天，2 月 28 天，闰年为 29 天）、一星期有 7 天……这些常识人人皆知。但计算机并不能自己分辨数据的对与错，只是依靠开发者提供的功能去执行或检查。PHP 中内置了日期检查函数，就是 checkdate()函数。checkdate()函数的语法如下：

```
bool checkdate(int month,int day,int year)
```

其中，month 的有效值为 1～12；day 的有效值为当月的最大天数，如 1 月为 31 天，2 月为 29 天（闰年）；year 的有效值从 1～32767。

【例 10.5】 本实例将观察使用 checkdate()函数的返回值，一个为正确的日期，一个为错误的日期，实例代码如下：**（实例位置：光盘\TM\sl\10\5）**

```
<?php
	$year = 2008;                                    //年份
	$month = 2;                                      //月份
	$day1 = 29;                                      //每月份的第几天
	$day2 = 30;                                      //每月份的天数
	var_dump(checkdate($month,$day1,$year));         //查看第一种情况的返回结果
	echo "<br>";
	var_dump(checkdate($month,$day2,$year));         //查看第二种情况的返回结果
?>
```

运行结果如图 10.5 所示。



图 10.5　使用 checkdate()函数验证日期

10.2.6　输出格式化的日期和时间

格式化时间函数 date()的语法在 10.2.1 节中已经讲解过，这里重点讲解 date()函数的参数 format 的

格式化选项，如表 10.4 所示。

表 10.4　参数 format 的格式化选项

参　　数	说　　明
a	小写的上午和下午值，返回值 am 或 pm
A	大写的上午和下午值，返回值 AM 或 PM
B	Swatch Internet 标准时间，返回值 000～999
d	月份中的第几天，有前导零的两位数字，返回值 01～31
D	星期中的第几天，文本格式，3 个字母，返回值 Mon～Sun
F	月份，完整的文本格式，返回值 January～December
h	小时，12 小时格式，没有前导零，返回值 1～12
H	小时，24 小时格式，没有前导零，返回值 0～23
i	有前导零的分钟数，返回值 00～59
I	判断是否为夏令时，返回值如果是夏令时为 1，否则为 0
j	月份中的第几天，没有前导零，返回值 1～31
l（L 的小写）	星期数，完整的文本格式，返回值 Sunday～Saturday
L	判断是否为闰年，返回值如果是闰年为 1，否则为 0
m	数字表示的月份，有前导零，返回值 01～12
M	3 个字母缩写表示的月份，返回值 Jan～Dec
n	数字表示的月份，没有前导零，返回值 1～12
O	与格林威治时间相差的小时数，如+0200
r	RFC 822 格式的日期，如 Thu，21 Dec 2000 16：01：07 +0200
s	秒数，有前导零，返回值 00～59
S	每月天数后面的英文后缀，两个字符，如 st、nd、rd 或者 th。可以和 j 一起使用
t	指定月份所应有的天数，28～31
T	本机所在的时区
U	从 UNIX 纪元（January 1 1970 00:00:00 GMT）开始至今的秒数
w	星期中的第几天，数字表示，返回值为 0~6
W	ISO-8601 格式年份中的第几周，每周从星期一开始
y	两位数字表示的年份，返回值如 88 或 08
Y	4 位数字完整表示的年份，返回值如 1998、2008
z	年份中的第几天，返回值 0～366
Z	时差偏移量的秒数。UTC 西边的时区偏移量总是负的，UTC 东边的时区偏移量总是正的，返回值-43200～43200

【例 10.6】 date()函数可以对 format 选项随意地组合。在本例中，既有单独输出一个参数的情况，也有输出多个参数的情况，最后还输出了转义字符。实例代码如下：**（实例位置：光盘\TM\sl\10\6）**

```
<?php
    echo "输出单个变量：".date("Y")."-".date("m")."-".date("d");          //输出单个参数
    echo "<p>";
    echo "输出组合变量：".date("Y-m-d");                                  //输出组合参数
    echo "<p>";
    echo "输出更详细的日期及时间：".date("Y-m-d H:i:s");                  //输出详细的日期和时间参数
```



```
    echo "<p>";
    echo "还可以更详细吗？？";
    echo date("l Y-m-d H:i:s T");                    //除了时间，再输出星期及所在时区
    echo "<p>";
    echo "输出转移字符：";
    echo date("\T\o\d\a\y \i\s \\t\h\e jS \o\f \y\e\a\\r");    //输出转移字符

?>
```

运行结果如图 10.6 所示。

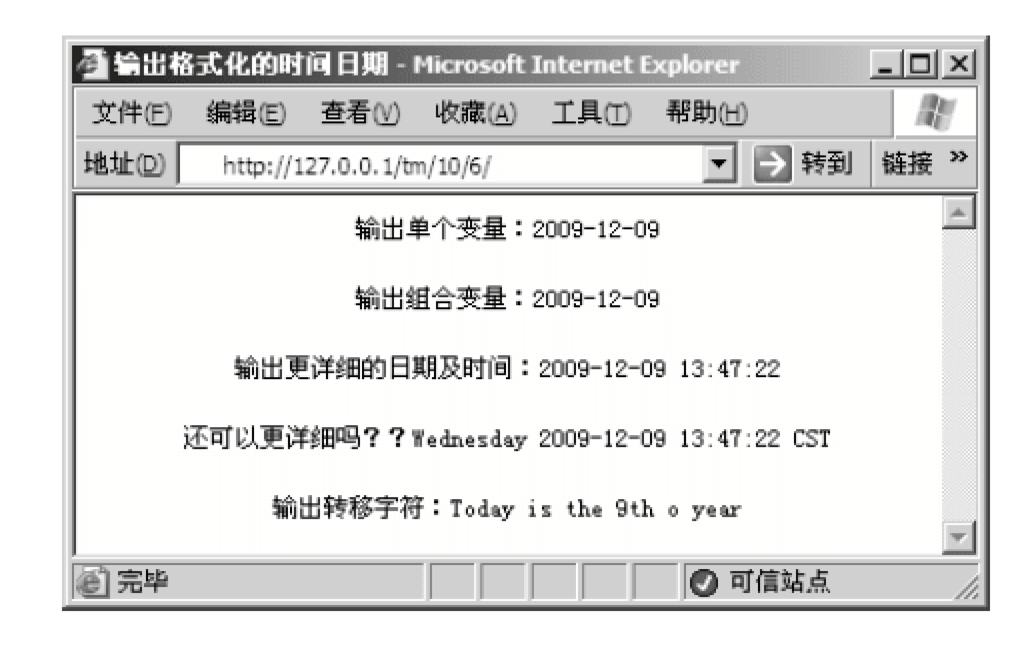


图 10.6　输出格式化的时间日期

10.2.7　显示本地化的日期和时间

不同的国家和地区，使用不同的时间、日期、货币的表示法和不同的字符集。如例 10.4 中的星期，在大多数西方国家都使用 Friday，但在以汉语为主的国家中，都使用星期五，虽然都是同一个含义，但表示的方式却不尽相同，这时就需要设置本地化环境。这里将使用 setlocale()函数和 strftime()函数来设置本地化环境和格式化输出日期和时间。下面分别对这两个函数进行介绍。

1．setlocale()函数

setlocale()函数可以改变 PHP 默认的本地化环境。

语法格式如下：

```
string setlocale(string category, string locale)
```

参数 category 的选项如表 10.5 所示。

表 10.5　category 参数选项及说明

参　　数	说　　明
LC_ALL	包含了下面所有的设置本地化规则
LC_COLLATE	字符串比较
LC_CTYPE	字符串分类和转换，如转换大小写
LC_MONETARY	本地化环境的货币形式
LC_NUMERIC	本地化环境的数值形式
LC_TIME	本地化环境的时间形式

参数 locale 如果为空，就会使用系统环境变量的 locale 或 lang 的值，否则就会应用 locale 参数所指定的本地化环境。如 en_US 为美国本地化环境，chs 则指简体中文，cht 为繁体中文。


说明

对于 Windows 平台的用户，可以登录 http://msdn.microsoft.com 来获取语言和国家（地区）的编码列表。如果是 UNIX/Linux 系统，则可以使用命令 locale–a 来确定所支持的本地化环境。

2. strftime()函数

strftime()函数根据本地化环境设置来格式化输出日期和时间。

语法格式如下：

```
string strftime(string format, int timestamp)
```

该函数返回用给定的字符串对参数 timestamp 进行格式化后输出的字符串。如果没有给出时间戳则用本地时间。月份、星期以及其他和语言有关的字符串写法和 setlocale 函数设置的当前区域有关。format 参数识别的转换标记如表 10.6 所示。

表 10.6　参数 format 识别的转换标记

参　数	说　明
%a	星期的简写
%A	星期的全称
%b	月份的简写
%B	月份的全称
%c	当前区域首选的日期时间表达
%C	世纪值（年份除以 100 后取整，范围从 00～99）
%d	月份中的第几天，十进制数字（范围从 01～31）
%D	和%m/%d/%y 一样
%e	月份中的第几天，十进制数字，一位的数字前会加上一个空格（范围从 1～31）
%g	和%G 一样，但没有世纪值
%G	4 位数的年份，符合 ISO 星期数（参见%V）。与%V 的格式和值一样，不同的是如果 ISO 星期数属于前一年或者后一年，则使用那一年
%h	和%b 一样
%H	24 小时制的十进制小时数（范围从 00～23）
%I	12 小时制的十进制小时数（范围从 00～12）
%j	年份中的第几天，十进制数（范围从 001～366）
%m	十进制月份（范围从 01～12）
%M	十进制分钟数
%n	换行符
%p	根据给定的时间值为 am 或 pm，或者当前区域设置中的相应字符串
%r	用 a.m 和 p.m 符号的时间
%R	24 小时符号的时间

续表

参　　数	说　　明
$S	十进制秒数
%t	制表符
%T	当前时间，和%H:%M:%S 一样
%u	星期几的十进制数表达[1,7]，1 表示星期一
%U	本年的第几周，从第一周的第一个星期天作为第一天开始
%V	本年第几周的 ISO 8601:1988 格式，范围从 01～53，第一周是本年第一个至少还有 4 天的星期，星期一作为每周的第一天（用%G 或者%g 作为指定时间戳相应周数的年份组成）
%W	本年的第几周数，从第一周的第一个星期一作为第一天开始
%w	星期中的第几天，星期天为 0
%x	当前区域首选的时间表示法，不包括时间
%X	当前区域首选的时间表示法，不包括日期
%y	没有世纪数的十进制年份（范围从 00～99）
%Y	包括世纪数的十进制年份
%Z（或%z）	时区名或缩写
%%	文字上的%字符

说明

对于 strftime()函数，可能不是所有的转换标记都被 C 库文件支持，这种情况下 PHP 的 strftime()也不支持。此外，不是所有的平台都支持负的时间戳，因此日期的范围可能限定在不早于 UNIX 纪元。这意味着，%e，%T，%R 和%D（可能更多）以及早于 Jan 1, 1970 的时间在 Windows、Linux 以及其他几个操作系统中无效。对于 Windows 系统，所支持的转换标记可在 MSDN 网站找到。

【例 10.7】 本实例分别使用 en_US、chs 和 cht 来输出今天是星期几，实例代码如下：（**实例位置：光盘\TM\sl\10\7**）

```
<?php
	setlocale(LC_ALL,"en_US");
	echo "美国格式：".strftime("Today is %A");
	echo "<p>";
	setlocale(LC_ALL,"chs");
	echo "中文简体格式：".strftime("今天是%A");
	echo "<p>";
	setlocale(LC_ALL,"cht");
	echo "<p>";
	echo "繁体中文格式：".strftime("今天是%A");
?>
```

运行结果如图 10.7 所示。



图 10.7　本地化日期


说明

因为本页面中的编码格式为 GB2312，所以最后繁体中文显示的日期为乱码，如果将编码格式改为 big5，繁体中文将显示出来，但其他文字则变为乱码。可以选择“查看”/“编码”命令，在弹出的菜单中选择“繁体中文(big5)”选项查看效果。


注意

如果在系统中没有安装各自的区域设置，是无法工作的。

10.2.8　将日期和时间解析为 UNIX 时间戳

PHP 中应用 strtotime()函数可将任何英文文本的日期和时间解析为 UNIX 时间戳，其值相对于 now 参数给出的时间，如果没有提供此参数则用系统当前时间。

语法：

```
int strtotime ( string time [, int now] )
```

该函数有两个参数。如果参数 time 的格式是绝对时间，则 now 参数不起作用；如果参数 time 的格式是相对时间，那么其对应的时间就是参数 now 来提供的，如果没有提供参数 now，对应的时间就为当前时间。如果解析失败返回 false。在 PHP 5.1.0 之前本函数在失败时返回-1。

【例 10.8】 本实例应用 strtotime()函数获取英文格式日期时间字符串的 UNIX 时间戳，并将部分时间输出。实例代码如下：**（实例位置：光盘\TM\sl\10\8）**

```
<?php
    echo strtotime ("now"), "\n";                                          //当前时间的时间戳
    echo "输出时间:".date("Y-m-d H:i:s",strtotime ("now")),"<br>";          //输出当前时间
    echo strtotime ("21 May 2009"), "\n";                                  //输出指定日期的时间戳
    echo "输出时间:".date("Y-m-d H:i:s",strtotime ("21 May 2009")),"<br>";  //输出指定日期的时间
    echo strtotime ("+3 day"), "\n";
    echo "输出时间:".date("Y-m-d",strtotime ("+3 day")),"<br>";
    echo strtotime ("+1 week")."<br>";
```

```
    echo strtotime ("+1 week 2 days 3 hours 4 seconds")."<br>";
    echo strtotime ("next Thursday")."<br>";
    echo strtotime ("last Monday"), "\n";
?>
```

运行结果如图 10.8 所示。

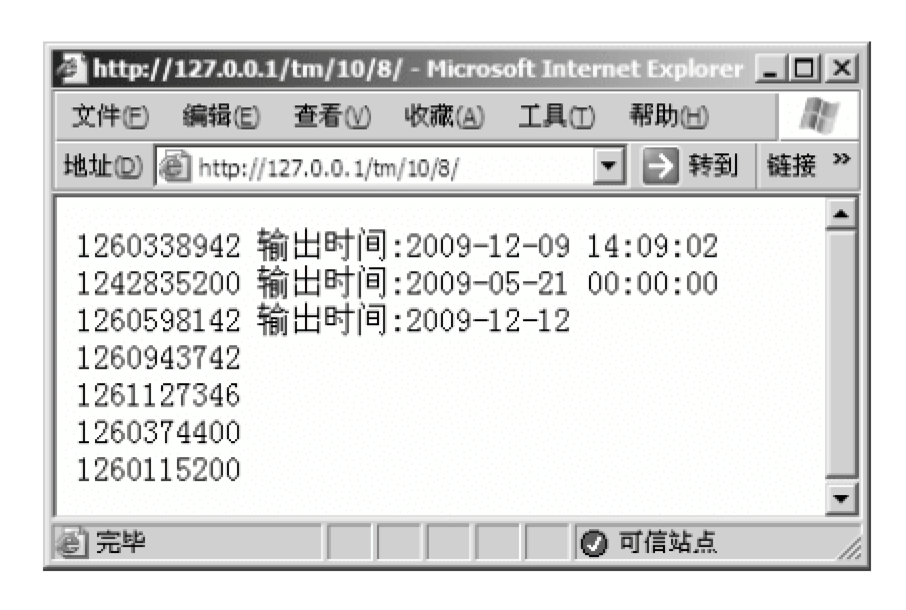


图 10.8　使用 strtotime()函数将日期和时间解析为 UNIX 时间戳

10.3　日期和时间的应用

视频讲解：光盘\TM\lx\10\日期和时间的应用.exe

本节中将介绍几个日期和时间的常用方法。

10.3.1　比较两个时间的大小

在实际开发中经常会对两个时间的大小进行判断，PHP 中的时间是不可以直接进行比较的。所以，首先要将时间解析为时间戳的格式，然后再进行比较。在 10.2.8 节中介绍的 strtotime()函数即可完成该操作。

【例 10.9】 本例先声明两个时间变量，然后使用 strtotime()函数对两个变量进行解析，再求差，最后根据差值输出结果。实例代码如下：（**实例位置：光盘\TM\sl\10\9**）

```
<?php
    $time1 = date("Y-m-d H:i:s");                         //获取当前时间
    $time2 = "2008-2-3 16:30:00";                         //给变量$time2 设置一个时间
    echo "变量\$time1 的时间为：".$time1."<br>";          //输出两个时间变量
    echo "变量\$time2 的时间为：".$time2."<p>";
    if(strtotime($time1) - strtotime($time2) < 0){        //对两个时间进行运算
        echo "\$time1 早于\$time2 ";                      //如果 time1–time2<0，说明 time1 的时间在前
    }else{
        echo "\$time2 早于$time1";                        //否则，说明 time2 的时间在前
    }
?>
```

运行结果如图 10.9 所示。



图 10.9　使用 strtotime()函数比较两个时间的大小

10.3.2　实现倒计时功能

【例 10.10】 除了可以比较两个日期的大小，还可以精确地计算出两个日期的差值。这里仍然使用 strtotime()函数，开发一个倒计时的小程序。实例代码如下：**（实例位置：光盘\TM\sl\10\10）**

```
<?PHP
    $time1 = strtotime(date( "Y-m-d H:i:s"));              //当前的系统时间
    $time2 = strtotime("2010-2-10 17:10:00");              //放假的时间
    $time3 = strtotime("2010-1-1");                        //2010 年元旦
   $sub1 = ceil(($time2 - $time1) / 3600);                 //(60 秒*60 分)秒/小时
    $sub2 = ceil(($time3 - $time1) / 86400);               //(60 秒*60 分*24 小时)秒/天
     echo "离放假还有<font color=red> $sub1 </font>小时!!!" ;
     echo "<p>";
     echo "离 2010 年元旦还有<font color=red>$sub2 </font>天!!!";
?>
```

说明

ceil()函数的格式为 float ceil(float value)，该函数为取整函数，返回不小于参数 value 值的最小整数。如果有小数部分，则进一位。应注意该函数的返回类型为 float 型，而不是整型。

运行结果如图 10.10 所示。



图 10.10　计算两个时间的差值

10.3.3　计算页面脚本的运行时间

在浏览网站时，经常会用到搜索引擎，在搜索信息时，细心的用户会发现，在搜索结果的最下方，



一般都有“搜索时间为 X 秒”的字样。

这里使用到了 microtime()函数，该函数返回当前 UNIX 时间戳和微秒数。返回格式为 msec sec 的字符串，其中 sec 是当前的 UNIX 时间戳，msec 为微秒数。

语法格式如下：

```
string microtime(void)
```

【例 10.11】 下面就来计算一下例 10.10 的运行时间。首先声明一个函数 run_time()，该函数返回当前的时间，精确到微秒。在 PHP 代码段运行之前先运行一次该函数，同时保存到变量$start_time 中，随后运行 PHP 代码段。当代码段运行完毕后再次调用 run_time()函数，同时保存到变量$end_time 中，这两个变量的差值就是该 PHP 代码段运行的时间。实例代码如下：（**实例位置：光盘\TM\sl\10\11**）

```
<?php
/*    声明 run_time 函数   */
function run_time(){
    list($msec, $sec) = explode(" ", microtime());          //使用 explode 函数返回两个变量
    return ((float)$msec + (float)$sec);                    //返回两个变量的和
}
        $start_time = run_time();                           //第一次运行 run_time()函数
/*    运行 PHP 代码段      */

       $time1 = strtotime(date( "Y-m-d H:i:s"));
       $time2 = strtotime("2010-2-10 17:10:00");
       $time3 = strtotime("2010-1-1");
        $sub1 = ceil(($time2 - $time1) / 3600);             //60 * 60，即每小时包含的秒数
       $sub2 = ceil(($time3 - $time1) / 86400);             //60 * 60 * 24，即每天包含的秒数
        echo "离放假还有<font color=red> $sub1 </font>小时!!!" ;
        echo "<p>";
        echo "离 2010 年元旦还有<font color=red>$sub2 </font>天!!!";
        $end_time = run_time();                             //再次运行 run_time()函数
?>
<p>
<!--   输出差值  -->
该示例的运行时间为<font color=blue> <?php echo ($end_time - $start_time); ?> </font>秒
```

代码说明：

☑ explode()函数。函数格式为 array explode(string separator, string string)。该函数的作用是将字符串（string）依照指定的字符串或字符（separator）切开，如果 separator 为空（""），那么函数将返回 false；如果 separator 所包含的值在 string 中找不到，那么函数将返回 string 单个元素的数组。

☑ list()函数。函数格式为 void list(mixed…)。该函数的作用是将数组中的值赋给一些变量（mixed）。

运行结果如图 10.11 所示。

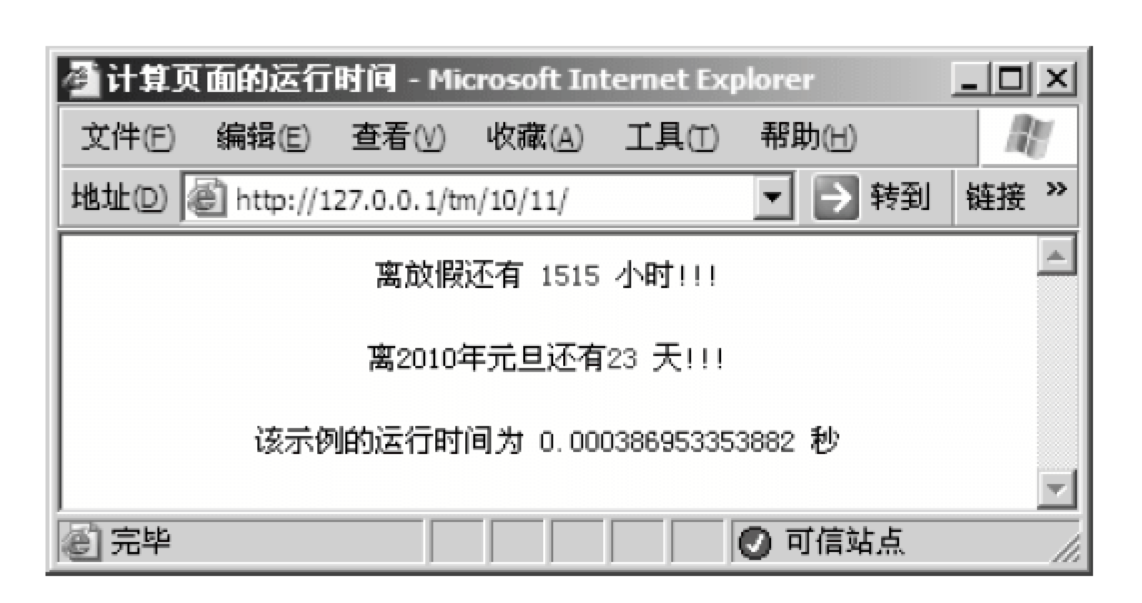


图 10.11　计算页面的运行时间

10.4　小　　结

本章介绍了 PHP 中常用的处理日期和时间的函数，主要分 3 个方面。首先介绍了系统的时区设置，然后介绍了 PHP 中内置的日期和时间函数，最后介绍了日期和时间的常用方法。希望通过本章的学习，读者可以熟练地使用 PHP 中的日期和时间函数，举一反三，实现更简单、更精妙的功能。

10.5　练习与实践

1．获取指定任意一天的时间，格式为 YYYY-MM-DD HH:MM:SS。**（答案位置：光盘\TM\sl\10\12）**

2．使用多种方法计算两个时间的差。**（答案位置：光盘\TM\sl\10\13）**

第 2 篇

核心技术

本篇介绍了 Cookie 与 Session、图形图像处理技术、文件系统、面向对象、PHP 加密技术、MySQL 数据库基础、phpMyAdmin 图形化管理工具、PHP 操作 MySQL 数据库、ADODB 类库、Zend Framework 框架等。掌握本篇内容后，能够开发数据库应用程序和一些中小型的热点模块。

第 11 章

Cookie 与 Session

（视频讲解：1 小时 6 分钟）

Cookie 和 Session 是两种不同的存储机制，前者是从一个 Web 页到下一个页面的数据传递方法，存储在客户端；后者是让数据在页面中持续有效的方法，存储在服务器端。可以说，掌握 Cookie 和 Session 技术，对于 Web 网站页面间信息传递的安全性是必不可少的。

通过阅读本章，您可以：

- 了解 Cookie 是什么以及 Cookie 能做什么
- 掌握如何创建 Cookie
- 掌握读取 Cookie 的方法
- 掌握删除 Cookie 的两种方法
- 了解 Cookie 的生命周期
- 了解 Session 是什么以及 Session 能做什么
- 掌握启动会话、注册会话、使用会话、删除会话的方法
- 掌握 Session 的高级应用

11.1　Cookie 管理

视频讲解：光盘\TM\lx\11\Cookie 管理.exe

Cookie 是在 HTTP 协议下，服务器或脚本可以维护客户工作站上信息的一种方式。Cookie 的使用很普遍，许多提供个人化服务的网站都是利用 Cookie 来区别不同用户，以显示与用户相应的内容，如 Web 接口的免费 E-mail 网站，就需要用到 Cookie。有效地使用 Cookie 可以轻松完成很多复杂任务。下面对 Cookie 的相关知识进行详细介绍。

11.1.1　了解 Cookie

本节首先简单介绍 Cookie 是什么以及 Cookie 能做什么。希望读者通过本节的学习对 Cookie 有一个明确的认识。

1．什么是 Cookie

Cookie 是一种在远程浏览器端存储数据并以此来跟踪和识别用户的机制。简单地说，Cookie 是 Web 服务器暂时存储在用户硬盘上的一个文本文件，并随后被 Web 浏览器读取。当用户再次访问 Web 网站时，网站通过读取 Cookies 文件记录这位访客的特定信息（如上次访问的位置、花费的时间、用户名和密码等），从而迅速作出响应，如在页面中不需要输入用户的 ID 和密码即可直接登录网站等。

文本文件的命令格式如下：

```
用户名@网站地址[数字].txt
```

举个简单的例子，如果用户的系统盘为 C 盘，操作系统为 Windows 2000/XP/2003，当使用 IE 浏览器访问 Web 网站时，Web 服务器会自动以上述的命令格式生成相应的 Cookies 文本文件，并存储在用户硬盘的指定位置，如图 11.1 所示。

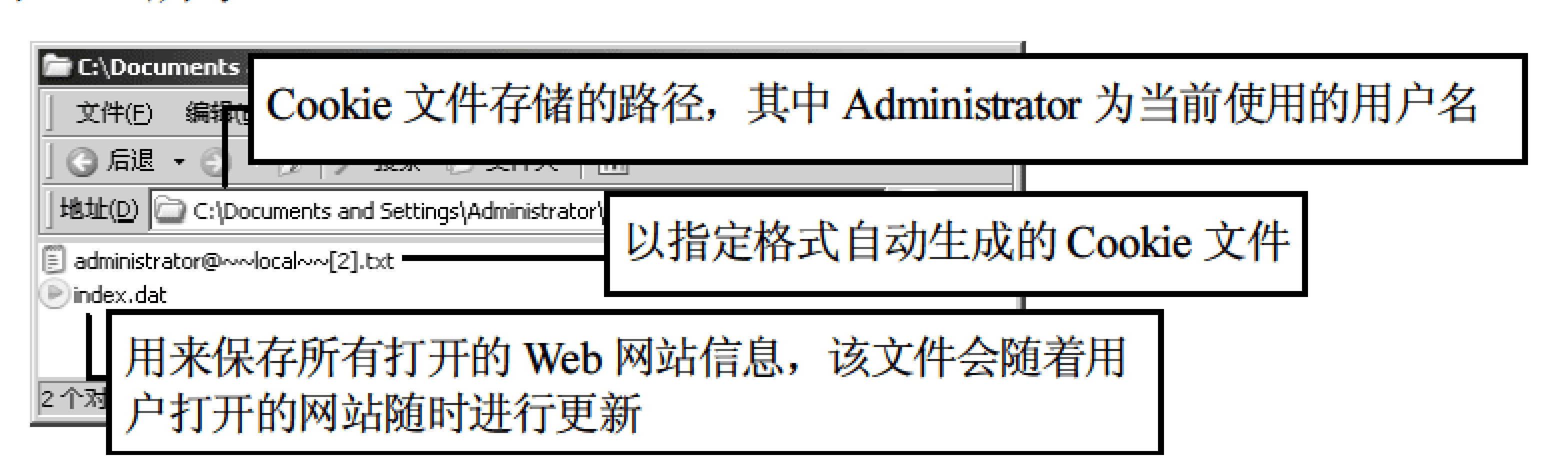
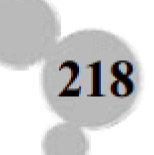


图 11.1　Cookie 文件的存储路径

注意

在 Cookies 文件夹下，每个 Cookie 文件都是一个简单而又普通的文本文件，而不是程序。Cookies 中的内容大多都经过了加密处理，因此，表面看来只是一些字母和数字组合，而只有服务器的 CGI 处理程序才知道它们真正的含义。



2．Cookie 的功能

Web 服务器可以应用 Cookies 包含信息的任意性来筛选并经常性维护这些信息，以判断在 HTTP 传输中的状态。Cookie 常用于以下 3 个方面：

☑ 记录访客的某些信息。如可以利用 Cookie 记录用户访问网页的次数，或者记录访客曾经输入过的信息，另外，某些网站可以使用 Cookie 自动记录访客上次登录的用户名。

☑ 在页面之间传递变量。浏览器并不会保存当前页面上的任何变量信息，当页面被关闭时页面上的所有变量信息将随之消失。如果用户声明一个变量 id=8，要把这个变量传递到另一个页面，可以把变量 id 以 Cookie 形式保存下来，然后在下一页通过读取该 Cookie 来获取变量的值。

☑ 将所查看的 Internet 页存储在 Cookies 临时文件夹中，可以提高以后浏览的速度。

注意

一般不要用 Cookie 保存数据集或其他大量数据。并非所有的浏览器都支持 Cookie，并且数据信息是以明文文本的形式保存在客户端计算机中，因此最好不要保存敏感的、未加密的数据，否则会影响网络的安全性。

11.1.2　创建 Cookie

在 PHP 中通过 setcookie()函数创建 Cookie。在创建 Cookie 之前必须了解的是，Cookie 是 HTTP 头标的组成部分，而头标必须在页面其他内容之前发送，它必须最先输出。若在 setcookie()函数前输出一个 HTML 标记或 echo 语句，甚至一个空行都会导致程序出错。

语法格式如下：

```
bool setcookie(string name[,string value[,int expire[, string path[,string domain[,int secure]]]]])
```

setcookie()函数的参数说明如表 11.1 所示。

表 11.1　setcookie()函数的参数说明

参　数	说　明	举　例
name	Cookie 的变量名	可以通过$_COOKIE["cookiename"]调用变量名为 cookiename 的 Cookie
value	Cookie 变量的值，该值保存在客户端，不能用来保存敏感数据	可以通过$_COOKIE["values"]获取名为 values 的值
expire	Cookie 的失效时间，expire 是标准的 UNIX 时间标记，可以用 time()函数或 mktime()函数获取，单位为秒	如果不设置 Cookie 的失效时间，那么 Cookie 将永远有效，除非手动将其删除
path	Cookie 在服务器端的有效路径	如果该参数设置为“/”，则它在整个 domain 内有效，如果设置为“/11”，它在 domain 下的/11 目录及子目录内有效。默认是当前目录

续表

参　数	说　明	举　例
domain	Cookie 有效的域名	如果要使 Cookie 在 mrbccd.com 域名下的所有子域都有效，应该设置为 mrbccd.com
secure	指明 Cookie 是否仅通过安全的 HTTPS，值为 0 或 1	如果值为 1，则 Cookie 只能在 HTTPS 连接上有效；如果值为默认值 0，则 Cookie 在 HTTP 和 HTTPS 连接上均有效

【例 11.1】 使用 setcookie()函数创建 Cookie，实例代码如下：（**实例位置：光盘\TM\sl\11\1**）

```
<?php
setcookie("TMCookie",'www.mrbccd.com');
setcookie("TMCookie", 'www.mrbccd.com', time()+60);                    //设置 Cookie 有效时间为 60 秒
//设置有效时间为 60 秒，有效目录为“/tm/”，有效域名为“mrbccd.com”及其所有子域名
setcookie("TMCookie", $value, time()+3600, "/tm/",". mrbccd.com", 1);
?>
```

运行本实例，在 Cookies 文件夹下会自动生成一个 Cookie 文件，名为 administrator@1[1].txt，Cookie 的有效期为 60 秒，在 Cookie 失效后，Cookies 文件自动删除。

11.1.3 读取 Cookie

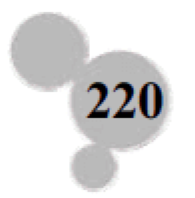
在 PHP 中可以直接通过超级全局数组$_COOKIE[]来读取浏览器端的 Cookie 值。

【例 11.2】 使用 print_r()函数读取 Cookie 变量，实例代码如下：（**实例位置：光盘\TM\sl\11\2**）

```
<?php
if(!isset($_COOKIE["visittime"])){                                      //检测 Cookie 文件是否存在，如果不存在
      setcookie("visittime",date("y-m-d H:i:s"));                       //设置一个 Cookie 变量
      echo "欢迎您第一次访问网站！";                                      //输出字符串
}else{                                                                  //如果 Cookie 存在
      setcookie("visittime",date("y-m-d H:i:s"),time()+60);             //设置保存 Cookie 失效时间的变量
       echo "您上次访问网站的时间为：".$_COOKIE["visittime"];              //输出上次访问网站的时间
      echo "<br>";                                                      //输出回车符
}
      echo "您本次访问网站的时间为：   ".date("y-m-d H:i:s");              //输出当前的访问时间
?>
```

在上面的代码中，首先使用 isset()函数检测 Cookie 文件是否存在，如果不存在，则使用 setcookie()函数创建一个 Cookie，并输出相应的字符串；如果 Cookie 文件存在，则使用 setcookie()函数设置 Cookie 文件失效的时间，并输出用户上次访问网站的时间。最后在页面输出本次访问网站的当前时间。

首次运行本实例，由于没有检测到 Cookie 文件，运行结果如图 11.2 所示。如果用户在 Cookie 设置到期时间（本例为 60 秒）前刷新或再次访问该实例，运行结果如图 11.3 所示。



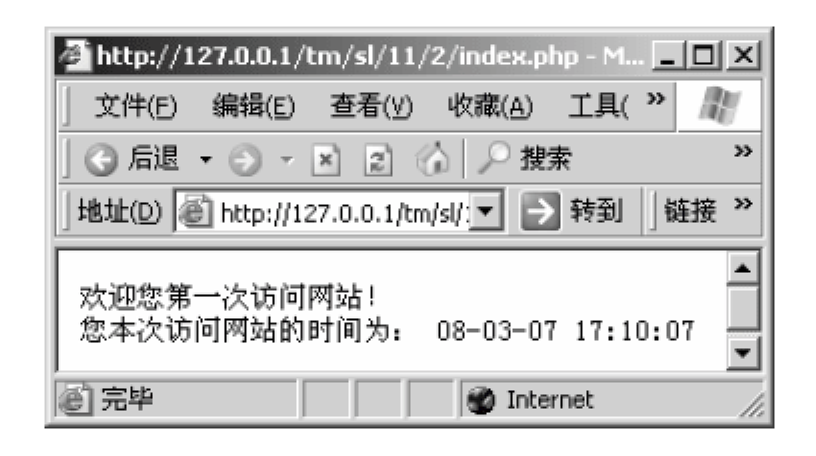


图 11.2　第一次访问网页的运行结果



图 11.3　刷新或再次访问本网页后的运行结果

注意

如果未设置 Cookie 的到期时间，则在关闭浏览器时自动删除 Cookie 数据。如果为 Cookie 设置了到期时间，浏览器将会记住 Cookie 数据，即使用户重启计算机，只要没到期，再访问网站时也会获得图 11.3 所示的数据信息。

11.1.4　删除 Cookie

当 Cookie 被创建后，如果没有设置它的失效时间，其 Cookie 文件会在关闭浏览器时被自动删除。如果要在关闭浏览器之前删除 Cookie 文件，方法有两种：一种是使用 setcookie()函数删除，另一种是在浏览器中手动删除 Cookie。下面分别进行介绍。

1．使用 setcookie()函数删除 Cookie

删除 Cookie 和创建 Cookie 的方式基本类似，删除 Cookie 也使用 setcookie()函数。删除 Cookie 只需要将 setcookie()函数中的第二个参数设置为空值，将第 3 个参数 Cookie 的过期时间设置为小于系统的当前时间即可。

例如，将 Cookie 的过期时间设置为当前时间减 1 秒，代码如下：

```
setcookie("name", "", time()-1);
```

在上面的代码中，time()函数返回以秒表示的当前时间戳，把过期时间减 1 秒就会得到过去的时间，从而删除 Cookie。



把过期时间设置为 0，可以直接删除 Cookie。

2．在浏览器中手动删除 Cookie

在使用 Cookie 时，Cookie 自动生成一个文本文件存储在 IE 浏览器的 Cookies 临时文件夹中。在浏览器中删除 Cookie 文件是非常便捷的方法。具体操作步骤如下：

启动 IE 浏览器，选择“工具”/“Internet 选项”命令，打开“Internet 选项”对话框，如图 11.4 所示。在“常规”选项卡中单击“删除 Cookies”按钮，将弹出如图 11.5 所示的“删除 Cookies”对话

框，单击“确定”按钮，即可成功删除全部 Cookie 文件。

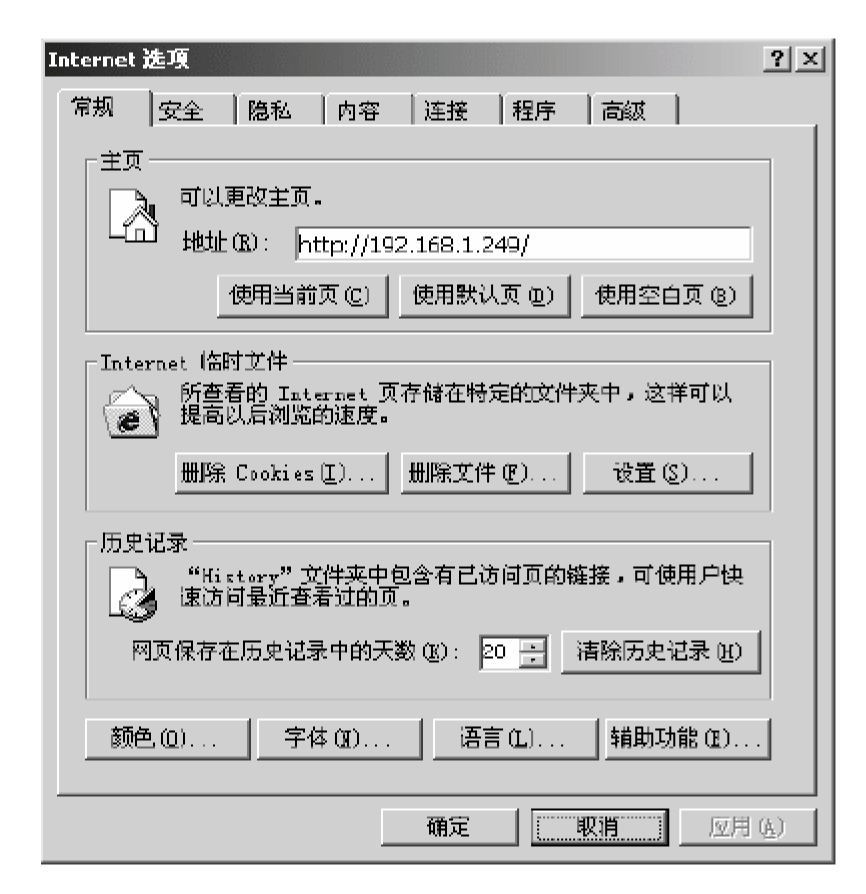


图 11.4 “Internet 选项”对话框



图 11.5 “删除 Cookies”对话框

11.1.5 Cookie 的生命周期

如果 Cookie 不设定时间，就表示它的生命周期为浏览器会话的期间，只要关闭 IE 浏览器，Cookie 就会自动消失。这种 Cookie 被称为会话 Cookie，一般不保存在硬盘上，而是保存在内存中。

如果设置了过期时间，那么浏览器会把 Cookie 保存到硬盘中，再次打开 IE 浏览器时会依然有效，直到它的有效期超时。

虽然 Cookie 可以长期保存在客户端浏览器中，但也不是一成不变的。因为浏览器最多允许存储 300 个 Cookie 文件，而且每个 Cookie 文件支持最大容量为 4KB；每个域名最多支持 20 个 Cookie，如果达到限制时，浏览器会自动地随机删除 Cookies。

11.2 Session 管理

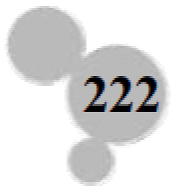
视频讲解：光盘\TM\lx\11\Session 管理.exe

对比 Cookie，会话文件中保存的数据是在 PHP 脚本中以变量的形式创建的，创建的会话变量在生命周期（20 分钟）中可以被跨页的请求所引用。另外，Session 是存储在服务器端的会话，相对安全，并且不像 Cookie 那样有存储长度的限制。

11.2.1 了解 Session

1. 什么是 Session

Session 译为“会话”，其本义是指有始有终的一系列动作/消息，如打电话时从拿起电话拨号到挂



断电话这一系列过程可以称为一个 Session。

在计算机专业术语中，Session 是指一个终端用户与交互系统进行通信的时间间隔，通常指从注册进入系统到注销退出系统所经过的时间。因此，Session 实际上是一个特定的时间概念。

2．Session 工作原理

当启动一个 Session 会话时，会生成一个随机且唯一的 session_id，也就是 Session 的文件名，此时 session_id 存储在服务器的内存中，当关闭页面时此 id 会自动注销，重新登录此页面，会再次生成一个随机且唯一的 id。

3．Session 的功能

Session 在 Web 技术中非常重要。由于网页是一种无状态的连接程序，因此无法得知用户的浏览状态。通过 Session 则可记录用户的有关信息，以供用户再次以此身份对 Web 服务器提交要求时作确认。例如，在电子商务网站中，通过 Session 记录用户登录的信息，以及用户所购买的商品，如果没有 Session，那么用户每进入一个页面都需要登录一次用户名和密码。

另外，Session 会话适用于存储信息量比较少的情况。如果用户需要存储的信息量相对较少，并且对存储内容不需要长期存储，那么使用 Session 把信息存储到服务器端比较合适。

11.2.2　创建会话

创建一个会话需要通过以下步骤：

```
启动会话→注册会话→使用会话→删除会话
```

1．启动会话

启动 PHP 会话的方式有两种：一种是使用 session_start()函数，另一种是使用 session_register()函数为会话登录一个变量来隐含地启动会话。



注意

通常，session_start()函数在页面开始位置调用，然后会话变量被登录到数据$_SESSION。

在 PHP 中有两种方法可以创建会话。

☑　通过 session_start ()函数创建会话。

语法格式如下：

```
bool session_start(void) ;
```



说明

使用 session_start()函数之前浏览器不能有任何输出，否则会产生类似于如图 11.6 所示的错误。

```
在session_start()函数前输出字符串，产生如下错误:
Warning: session_start() [function.session-start]: Cannot send session
cookie - headers already sent by (output started at F:\AppServ\www\TM\SL\11\4
\default.php:2) in F:\AppServ\www\TM\SL\11\4\default.php on line 3
```

图 11.6　在使用 session_start()函数前输出字符串产生的错误

☑ 通过 session_register()函数创建会话。

session_register()函数用来为会话登录一个变量来隐含地启动会话，但要求设置 php.ini 文件的选项，将 register_globals 指令设置为 on，然后重新启动 Apache 服务器。

注意

使用 session_register()函数时，不需要调用 session_start()函数，PHP 会在注册变量之后隐含地调用 session_start()函数。

2．注册会话

会话变量被启动后，全部保存在数组$_SESSION 中。通过数组$_SESSION 创建一个会话变量很容易，只要直接给该数组添加一个元素即可。

例如，启动会话，创建一个 Session 变量并赋予空值，代码如下：

```
<?php
session_start();                                        //启动 Session
$_SESSION["admin"] = null;                              //声明一个名为 admin 的变量，并赋空值
?>
```

3．使用会话

首先需要判断会话变量是否有一个会话 ID 存在，如果不存在，就创建一个，并且使其能够通过全局数组$_SESSION 进行访问。如果已经存在，则将这个已注册的会话变量载入以供用户使用。

例如，判断存储用户名的 Session 会话变量是否为空，如果不为空，则将该会话变量赋给$myvalue，代码如下：

```
<?php
if ( !empty ( $_SESSION['session_name']))               //判断用于存储用户名的 Session 会话变量是否为空
      $myvalue = $_SESSION['session_name'] ;            //将会话变量赋给一个变量$myvalue
?>
```

4．删除会话

删除会话的方法主要有删除单个会话、删除多个会话和结束当前会话 3 种，下面分别进行介绍。

（1）删除单个会话

删除会话变量，同数组的操作一样，直接注销$_SESSION 数组的某个元素即可。

例如，注销$_SESSION['user']变量，可以使用 unset()函数，代码如下：

```
unset ( $_SESSION['user'] ) ;
```



注意

使用 unset()函数时，要注意$_SESSION 数组中某元素不能省略，即不可以一次注销整个数组，这样会禁止整个会话的功能，如 unset($_SESSION) 函数会将全局变量$_SESSION 销毁，而且没有办法将其恢复，用户也不能再注册$_SESSION 变量。如果要删除多个或全部会话，可采用下面的两种方法。

（2）删除多个会话

如果想要一次注销所有的会话变量，可以将一个空的数组赋值给$_SESSION，代码如下：

```
$_SESSION = array() ;
```

（3）结束当前会话

如果整个会话已经结束，首先应该注销所有的会话变量，然后使用 session_destroy()函数清除结束当前的会话，并清空会话中的所有资源，彻底销毁 Session，代码如下：

```
session_destroy() ;
```

11.2.3　Session 设置时间

在大多数论坛中都可在登录时对登录时间进行选择，如保存一个星期、保存一个月等。这时就可以通过 Cookie 设置登录的失效时间。

1．客户端没有禁止 Cookie

（1）使用 session_set_cookie_params()设置 Session 的失效时间，此函数是 Session 结合 Cookie 设置失效时间，如要让 Session 在 1 分钟后失效，实例关键代码如下：**（实例位置：光盘\TM\sl\11\3）**

```
<?php
$time = 1 * 60;                                    //设置 Session 失效时间
session_set_cookie_params($time);                  //使用函数
session_start();                                   //初始化 Session
$_SESSION[username] = 'mr';
?>
```



注意

session_set_cookie_params()必须在 session_start()之前调用。



说明

不推荐使用此函数，此函数在一些浏览器上会出现问题。所以一般手动设置失效时间。

（2）使用 setcookie()函数可对 Session 设置失效时间，如让 Session 在 1 分钟后失效，实例关键代码如下：**（实例位置：光盘\TM\sl\11\4）**

```
<?php
session_start();
$time = 1 * 60;                                                    //给出 Session 失效时间
setcookie(session_name(),session_id(),time()+$time,"/");          //使用 setcookie()手动设置 Session 失效时间
$_SESSION['user'] = "mr";
?>
```



session_name 是 Session 的名称，session_id 是判断客户端用户的标识，因为 session_id 是随机产生的唯一名称，所以 Session 是相对安全的。失效时间和 Cookie 的失效时间一样，最后一个参数为可选参数，是放置 Cookie 的路径。

2．客户端禁止 Cookie

当客户端禁用 Cookie 时，Session 页面间传递会失效，可以将客户端禁止 Cookie 想象成一家大型连锁超市，如果在其中一家超市内办理了会员卡，但是超市之间并没有联网，那么会员卡就只能在办理的那家超市使用。解决这个问题有 4 种方法：

（1）在登录之前提醒用户必须打开 Cookie，这是很多论坛的做法。

（2）设置 php.ini 文件中的 session.use_trans_sid = 1，或者编译时打开-enable-trans-sid 选项，让 PHP 自动跨页面传递 session_id。

（3）通过 GET 方法，隐藏表单传递 session_id。

（4）使用文件或者数据库存储 session_id，在页面间传递中手动调用。

第二种情况不作详细讲解，因为用户不能修改服务器中的 php.ini 文件。第 3 种情况我们就不可以使用 Cookie 设置保存时间，但是登录情况没有变化。第 4 种也是最为重要的一种，在开发企业级网站时，如果遇到 Session 文件使服务器速度变慢，就可以使用。在 Session 高级应用中会作详细解说。

第 3 种情况使用 GET 方式传输，实例关键代码如下：**（实例位置：光盘\TM\sl\11\5）**

```
<form id="form1" name="form1" method="post" action="common.php?<?=session_ name();?>= <?=session_id(); ?>">
```

接收页面头部详细代码：

```
<?php
$sess_name = session_name();                          //取得 Session 名称
$sess_id = $_GET[$sess_name];                         //取得 session_id GET 方式
session_id($sess_id);                                 //关键步骤
session_start();
$_SESSION['admin'] = 'mrsoft';
?>
```

运行结果如图 11.7 所示。

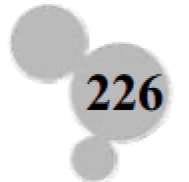




图 11.7　使用 GET 方式传递 session_id



说明

Session 原理为请求该页面之后会产生一个 session_id，如果这个时候禁止了 Cookie 就无法传递 session_id，在请求下一个页面时将会重新产生一个 session_id，这样就造成了 Session 在页面间传递失效。

11.2.4　通过 Session 判断用户的操作权限

在大多网站的开发过程中，需要对管理员和普通用户对操作网站的权限进行划分。下面通过具体的实例进行讲解。

首先通过用户登录页面提交的用户名来验证用户操作网站的权限。

【例 11.3】 本实例通过 Session 技术实现如何判断用户的操作权限。具体开发步骤如下：**（实例位置：光盘\TM\sl\11\6）**

（1）设计登录页面，添加一个表单 form1，应用 POST 方法进行传参，action 指向的数据处理页为 default.php，添加一个用户名文本框并命名为 user，添加一个密码域文本框并命名为 pwd，关键代码如下：

```
<form name="form1" method="post" action="default.php">
  <table width="521" height="394" border="0" cellpadding="0" cellspacing="0">
    <tr>
      <td valign="top" background="images/login.jpg">
        <table width="521" border="0" cellspacing="0" cellpadding="0">
          <tr>
            <td height="24" align="right">用户名：</td>
            <td height="24" align="left"><input name="user" type="text" id="user" size="20"></td>
          </tr>
          <tr>
            <td height="24" align="right">密  码：</td>
            <td height="24" align="left"><input name="pwd" type="password" id="pwd" size="20"></td>
          </tr>
```

```
            <tr align="center">
              <td height="24" colspan="2"><input type="submit" name="Submit" value="提交" onClick=
"return check(form);"><input type="reset" name="Submit2" value="重填"></td>
            </tr>
            <tr>
              <td height="76" align="right"><span class="style1">超级用户：tsoft<br>
  密    码：111  </span></td>
              <td><span class="style1">普通用户：zts<br>密    码：000</span></td>
            </tr>
        </table>
        </td>
      </tr>
  </table>
</form>
```

（2）在“提交”按钮的单击事件下，调用自定义函数 check()来验证表单元素是否为空。自定义函数 check()的代码如下：

```
<script language="javascript">
      function check(form){
            if(form.user.value==""){
                  alert("请输入用户名");form.user.focus();return false;
            }
            if(form.pwd.value==""){
                  alert("请输入密码");form.pwd.focus();return false;
            }
            form.submit();
      }
</script>
```

（3）提交表单元素到数据处理页 default.php，首先使用 session_start()函数初始化 Session 变量，然后通过 POST 方法接收表单元素的值，将获取的用户名和密码分别赋给 Session 变量，代码如下：

```
<?php
session_start();
$_SESSION[user]=$_POST[user];
$_SESSION[pwd]=$_POST[pwd];
?>
```

（4）为了防止其他用户非法登录本系统，使用 if 条件语句对 Session 变量值进行判断，代码如下：

```
<?php
if($_SESSION[user]==""){                                    //如果用户名为空，则弹出提示，并跳转到登录页
   echo "<script language='javascript'>alert('请通过正确的途径登录本系统！');history.back();</script>";
}
?>
```

（5）在数据处理页 default.php 的导航栏处添加如下代码：

```
<TABLE align="center" cellPadding=0 cellSpacing=0 >
    <TR align="center" valign="middle">
```



```
        <TD style="WIDTH: 140px; COLOR: red;">当前用户: 
        <!-- ------------------------------------------输出当前登录的用户级别---------------------------------------- -->
        <?php if($_SESSION[user]=="tsoft"&& $_SESSION[pwd]=="111"){echo "管理员";}else{echo "普通用户";}
?>  
        </TD>
        <TD width="70"><a href="default.php">博客首页</a></TD>
        <TD width="70">| <a href="default.php" >我的文章</a></TD>
        <TD width="70">| <a href="default.php" >我的相册</a></TD>
        <TD width="70">| <a href="default.php">音乐在线</a></TD>
        <TD width="70">| <a href="default.php">修改密码</a></TD>
        <?php
         if($_SESSION[user]=="tsoft"&& $_SESSION[pwd]=="111"){       //如果当前用户是管理员
         ?>
        <!-- ------------------------------如果当前用户是管理员，则输出“用户管理”链接--------------------------- -->
        <TD width="70">| <a href="default.php">用户管理</a></TD>
        <?php
           }
         ?>
    </TR>
</TABLE>
```

（6）在 default.php 页面添加“注销用户”超链接页 safe.php，该页代码如下：

```
<?php
session_start();                                   //初始化 Session
unset($_SESSION['user']);                          //删除用户名会话变量
unset($_SESSION['pwd']);                           //删除密码会话变量
session_destroy();                                 //删除当前所有的会话变量
header("location:index.php");                      //跳转到博客用户登录页
?>
```

（7）运行本实例，在博客用户登录页面输入用户名和密码，以超级用户的身份登录网站，运行结果如图 11.8 所示。以普通用户身份登录网站的运行结果如图 11.9 所示。



图 11.8　超级用户登录网站的运行结果

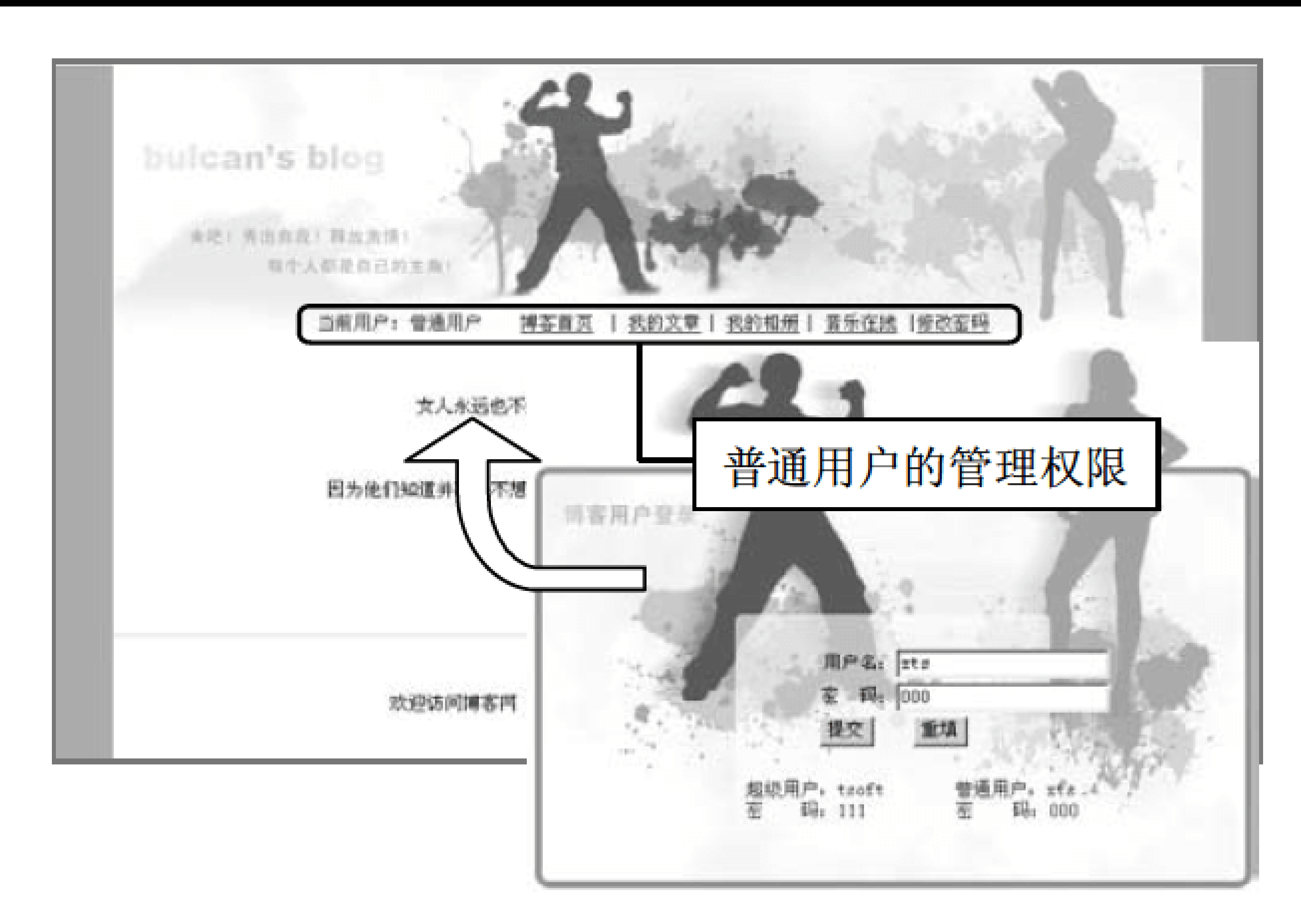


图 11.9　普通用户登录网站的运行结果

11.3　Session 高级应用

视频讲解：光盘\TM\lx\11\Session 高级应用.exe

11.3.1　Session 临时文件

在服务器中，如果将所有用户的 Session 都保存到临时目录中，会降低服务器的安全性和效率，打开服务器存储的站点会非常慢。

【例 11.4】 使用 PHP 函数 session_save_path()存储 Session 临时文件，可缓解因临时文件的存储导致服务器效率降低和站点打开缓慢的问题。实例代码如下：（**实例位置：光盘\TM\sl\11\7**）

```
<?php
$path = './tmp/';                               //设置 Session 存储路径
session_save_path($path);
session_start();                                //初始化 Session
$_SESSION[username] = true;
?>
```



注意

session_save_path()函数应在 session_start()函数之前调用。

11.3.2　Session 缓存

Session 的缓存是将网页中的内容临时存储到 IE 客户端的 Temporary Internet Files 文件夹下，并且



可以设置缓存的时间。当第一次浏览网页后，页面的部分内容在规定的时间内就被临时存储在客户端的临时文件夹中，这样在下次访问这个页面时，就可以直接读取缓存中的内容，从而提高网站的浏览效率。

Session 缓存的完成使用的是 session_cache_limiter()函数，其语法如下：

```
string session_cache_limiter ( [string cache_limiter])
```

参数 cache_limiter 为 public 或 private。同时 Session 缓存并不是指在服务器端而是客户端缓存，在服务器中没有显示。

缓存时间的设置，使用的是 session_cache_expire()函数，其语法如下：

```
int session_cache_expire ( [int new_cache_expire])
```

参数 cache_expire 是 Session 缓存的时间数字，单位是分钟。



注意

这两个 Session 缓存函数必须在 session_start()调用之前使用，否则出错。

【例 11.5】 下面通过实例了解 Session 缓存页面过程，实例代码如下：（**实例位置：光盘\TM\sl\11\8**）

```
<?php
session_cache_limiter('private');
$cache_limit = session_cache_limiter();                  //开启客户端缓存
session_cache_expire(30);
$cache_expire = session_cache_expire();                  //设定客户端缓存时间
session_start();
?>
```

运行结果如图 11.10 所示。



图 11.10　Session 客户端缓存

11.3.3　Session 数据库存储

虽然通过改变 Session 存储文件夹使 Session 不至于将临时文件夹填满而造成站点瘫痪，但是可以计算一下如果一个大型网站一天登录 1000 人，一个月登录了 30000 人，这时站点中存在 30000 个 Session 文件，要在这 30000 个文件中查询一个 session_id 应该不是件轻松的事情，那么这时就可以应用 Session

数据库存储，也就是 PHP 中的 session_set_save_handler()函数。

语法格式如下：

```
bool session_set_save_handler ( string open, string close, string read, string write, string destroy, string gc)
```

session_set_save_handler()函数的参数说明如表 11.2 所示。

表 11.2　session_set_save_handler()函数的参数说明

参　　数	说　　明
open(save_path,session_name)	找到 Session 存储地址，取出变量名称
close()	不需要参数，关闭数据库
read(key)	读取 Session 键值，key 对应 session_id
write(key,data)	其中 data 对应设置的 Session 变量
destroy(key)	注销 Session 对应 Session 键值
gc(expiry_time)	清除过期 Session 记录

一般应用参数直接使用变量，但是此函数中参数为 6 个函数，而且在调用时只是调用函数名称的字符串，下面将分别讲解这 6 个参数（函数），最后将把这些封装进类中，等学习完面向对象编程后就会有一个非常清晰的印象。

（1）封装 session_open()函数，连接数据库，代码如下：

```
function _session_open($save_path,$session_name)
{
	global $handle;
	$handle = mysql_connect('localhost','root','root') or die('数据库连接失败');		//连接 MySQL 数据库
	mysql_select_db('db_database11',$handle) or die('数据库中没有此库名');		//找到数据库
	return(true);
}
```

说明

$save_path 和$session_name 两个参数并没有用到，在这里可以将其去掉，但还是建议读者输入，因为一般使用都是存在这两个变量的，应该养成一个好的习惯。

（2）封装 session_close()函数，关闭数据库连接，代码如下：

```
function _session_close()
{
	global $handle;
	mysql_close($handle);
	return(true);
}
```

说明

在这个参数中不需要任何参数，所以不论是 Session 存储到数据库还是文件中，只需返回 true 即可。但是如果是 MySQL 数据库，最好是将数据库关闭，以保证以后不会出现麻烦。

（3）封装 session_read()函数，在函数中设定当前时间的 UNIX 时间戳，根据$key 值查找 Session 名称及内容，代码如下：

```
function _session_read($key)
{
	global $handle;                                         //全局变量$handle 连接数据库
	$time = time();                                         //设定当前时间
	$sql = "select session_data from tb_session where session_key = '$key' and session_time > $time";
	$result = mysql_query($sql,$handle);
	$row = mysql_fetch_array($result);
	if ($row){
		return($row['session_data']);                       //返回 Session 名称及内容
	}else{
		return(false);
	}
}
```

说明

存储进数据库中的 session_expiry 是 UNIX 时间戳。

（4）封装 session_write()函数，函数中设定 Session 失效时间，查找到 Session 名称及内容，如果查询结果为空，则将页面中 Session 根据 session_id、session_name、失效时间插入数据库中；如果查询结果不为空，则根据$key 修改数据库中 Session 存储信息，返回执行结果，代码如下：

```
function _session_write($key,$data)
{
	global $handle;
	$time = 60*60;                                                              //设置失效时间
	$lapse_time = time() + $time;                                               //得到 UNIX 时间戳
	$sql = "select session_data from tb_session where session_key = '$key' and session_time > $lapse_time";
	$result = mysql_query($sql,$handle);
	if (mysql_num_rows($result) == 0 ) {                                        //没有结果
		$sql = "insert into tb_session values('$key','$data',$lapse_time)";     //插入数据库 sql 语句
		$result = mysql_query($sql,$handle);
	}else{
		$sql = "update tb_session set session_key = '$key',session_data = '$data',session_time = 
$lapse_time where session_key = '$key'";                                        //修改数据库 sql 语句
		$result = mysql_query($sql,$handle);
	}
	return($result);
}
```

（5）封装 session_destroy()函数，根据$key 值将数据库中 Session 删除，代码如下：

```
function _session_destroy($key)
{
	global $handle;
	$sql = "delete from tb_session where session_key = '$key'";                 //删除数据库 sql 语句
```

```
	$result = mysql_query($sql,$handle);
	return($result);
}
```

（6）封装 session_gc()函数，根据给出的失效时间删除过期 Session，代码如下：

```
function _session_gc($expiry_time)
{
	global $handle;
	$lapse_time = time();                                               //将参数$expiry_time 赋值为当前时间戳
	$sql = "delete from tb_session where expiry_time < $lapse_time";   //删除数据库 sql 语句
	$result = mysql_query($sql,$handle);
	return($result);
}
```

以上为 session_set_save_handler()函数的 6 个参数（函数）。

【例 11.6】 下面通过函数 session_set_save_handler()实现 Session 存储数据库。实例代码如下：**（实例位置：光盘\TM\sl\11\9）**

```
session_set_save_handler('_session_open','_session_close','_session_read','_session_write','_session_destroy',
'_session_gc');
session_start();
//下面为我们定义的 Session
$_SESSION['user'] = 'mr';
$_SESSION['pwd'] = 'mrsoft';
```

现在可以查看数据库中表 Session 的内容，如图 11.11 所示。

←T→	session_key	session_data	session_time
☐ ✎ ✕	f5e66a38742eb37743f20d353f71e0c4	user|s:2:"mr";pwd|s:6:"mrsoft";SID|b:1;	1260846901

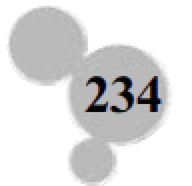
图 11.11　数据库存储 Session

11.4　小　　结

本章通过简短的篇幅让读者了解 Cookie 及 Session 是什么、能做些什么。下面总结了 Session 和 Cookie 有什么不同。

Session 和 Cookie 最大的区别是：Session 是将 Session 的信息保存在服务器上，并通过一个 Session ID 来传递客户端的信息，同时服务器接收到 Session ID 后，根据这个 ID 来提供相关的 Session 信息资源；Cookie 是将所有的信息以文本文件的形式保存在客户端，并由浏览器进行管理和维护。

由于 Session 为服务器存储，所以远程用户无法修改 Session 文件的内容。而 Cookie 为客户端存储，所以 Session 要比 Cookie 安全得多。当然使用 Session 还有很多优点，如控制容易，可以按照用户自定义存储等（存储于数据库）。



11.5　练习与实践

1．开发一个“试用版学习资源网”，当用户登录后，使用 Cookie 限制用户访问网站的时间，在页面停留 30 秒后，网站将提示“您在本网站停留的时间已经超过我们限制的时间，系统将在 5 秒钟后退出登录!!谢谢!请稍等…”。**（答案位置：光盘\TM\sl\11\10）**

2．使用 Session 技术实现聊天室换肤功能。**（答案位置：光盘\TM\sl\11\11）**

3．将 Session 数据库存储使用面向对象方法封装成类。**（答案位置：光盘\TM\sl\11\12）**

第12章

图形图像处理技术

（视频讲解：51 分钟）

由于有 GD 库的强大支持，PHP 的图像处理功能可以说是 PHP 的一个强项，便捷易用、功能强大。另外，PHP 图形化类库——Jpgraph 也是一款非常好用和强大的图形处理工具，可以绘制各种统计图和曲线图，也可以自定义设置颜色和字体等元素。

图像处理技术中的经典应用是绘制饼形图、柱形图和折线图，这是对数据进行图形化分析的最佳方法。本章将分别对 GD2 函数及 Jpgraph 类库进行详细讲解。

通过阅读本章，您可以：

- 了解、熟悉 GD 库的加载
- 掌握 Jpgraph 的安装与配置
- 熟练使用 GD2 函数创建图像
- 熟练使用 Jpgraph 类库创建柱形图
- 熟练使用 Jpgraph 类库创建折线图
- 熟练使用 Jpgraph 类库创建 3D 饼形图

12.1　在 PHP 中加载 GD 库

视频讲解：光盘\TM\lx\12\在 PHP 中加载 GD 库.exe

GD 库是一个开放的动态创建图像、源代码公开的函数库，可以从官方网站 http://www.boutell.com/gd 处下载。目前，GD 库支持 GIF、PNG、JPEG、WBMP 和 XBM 等多种图像格式，用于对图像的处理。

GD 库在 PHP 5 中是默认安装的，但要激活 GD 库，必须修改 php.ini 文件。将该文件中的“;extension=php_gd2.dll”选项前的分号“;”删除，如图 12.1 所示，保存修改后的文件并重新启动 Apache 服务器即可生效。

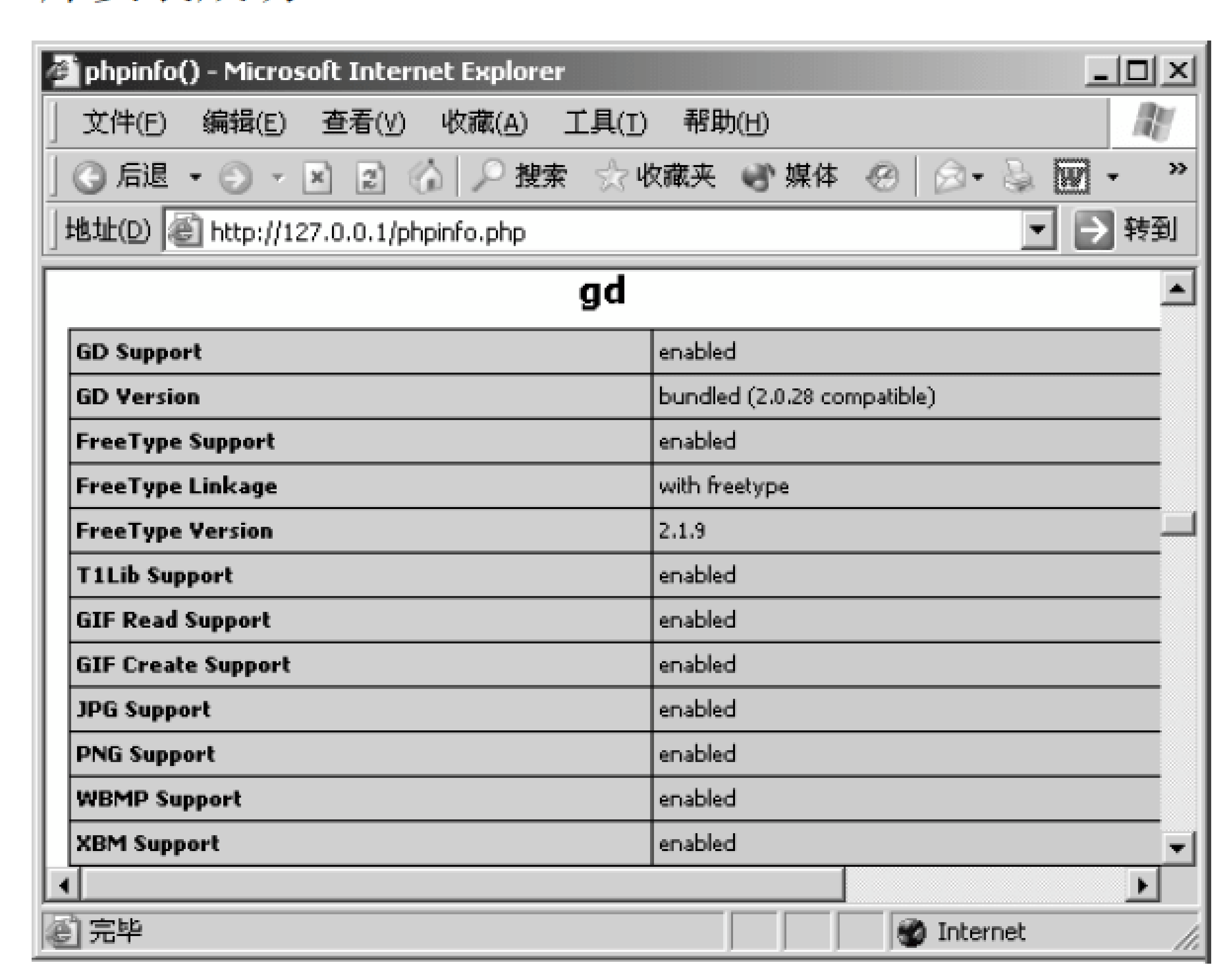
在成功加载 GD2 函数库后，可以通过 phpinfo()函数来获取 GD2 函数库的安装信息，验证 GD 库是否安装成功。

在 IE 浏览器的地址栏中输入“127.0.0.1/phpinfo.php”并按 Enter 键，在打开的页面中检索到如图 12.2 所示的 GD 库的安装信息，即说明 GD 库安装成功。

php.ini - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
extension=php_gd2.dll
```

图 12.1　加载 GD2 函数库



GD Support	enabled
GD Version	bundled (2.0.28 compatible)
FreeType Support	enabled
FreeType Linkage	with freetype
FreeType Version	2.1.9
T1Lib Support	enabled
GIF Read Support	enabled
GIF Create Support	enabled
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled
XBM Support	enabled

图 12.2　GD2 函数库的安装信息

说明

如果使用集成化安装包来配置 PHP 的开发环境，就不必担心这个问题，因为在集成化安装包中，默认 GD2 函数库已经被加载。

说明

Linux 和 Windows 系统下都可以使用 GD 库，函数也是完全一致，但是图形的坐标会发生偏移，如果两个系统互相移植，则必须重新查看界面。



12.2　Jpgraph 的安装与配置

视频讲解：光盘\TM\lx\12\Jpgraph 的安装与配置.exe

Jpgraph 这个强大的绘图组件能根据用户的需要绘制任意图形。只需要提供数据，就能自动调用绘图函数的过程，把处理的数据输入自动绘制。Jpgraph 提供了多种方法创建各种统计图，包括折线图、柱形图和饼形图等。Jpgraph 是一个完全使用 PHP 语言编写的类库，并可以应用在任何 PHP 环境中。

12.2.1　Jpgraph 的安装

Jpgraph 可以从其官方网站 http://www.aditus.nu/jpgraph/下载。注意 Jpgraph 支持 PHP 4.3.1 以上和 PHP 5 两种版本的图形库，应选择合适的 Jpgraph 版本下载，目前最新的版本是 2.3。

Jpgraph 的安装方法非常简单，文件下载后，安装步骤如下：

（1）将压缩包下的全部文件解压到一个文件夹中，如 F:\AppServ\www\jpgraph。

（2）打开 PHP 的安装目录，编辑 php.ini 文件并修改其中的 include_path 参数，在其后增加前面的文件夹名，如 include_path = ".;F:\AppServ\www\jpgraph"。

（3）重新启动 Apache 服务器即可生效。

注意

Jpgraph 需要 GD 库的支持。如果用户希望 Jpgraph 类库仅对当前站点有效，只需将 Jpgraph 压缩包下的 src 文件夹中的全部文件复制到网站所在目录的文件夹中，使用时调用 src 文件夹下的指定文件即可。这些内容在后面的典型实例中将具体讲解。

12.2.2　Jpgraph 的配置

Jpgraph 提供了一个专门用于配置 Jpgraph 类库的文件 jpg-config.inc.php。在使用 Jpgraph 前，可以通过修改文本文件来完成 Jpgraph 的配置。

jpg-config.inc.php 文件的配置需修改以下两项。

☑　支持中文的配置

Jpgraph 支持的中文标准字体可以通过修改 CHINESE_TTF_FONT 的设置来完成。

```
DEFINE('CHINESE_TTF_FONT','bkai00mp.ttf');
```

☑　默认图片格式的配置

根据当前 PHP 环境中支持的图片格式来设置默认的生成图片的格式。Jpgraph 默认图片格式的配置可以通过修改 DEFAULT_GFORMAT 的设置来完成。默认值 auto 表示 Jpgraph 将依次按照 PNG、GIF 和 JPEG 的顺序来检索系统支持的图片格式。

```
DEFINE("DEFAULT_GFORMAT","auto");
```


注意

如果用户使用的为 Jpgraph 2.3 版本，那么不需要重新进行配置。

12.3　图形图像的典型应用

视频讲解：光盘\TM\lx\12\图形图像的典型应用.exe

网页中如果没有丰富多彩的图形图像总是缺少生气，漂亮的图形图像能让整个网页看起来更富有吸引力，使许多文字难以表达的思想一目了然，并且可以清晰地表达出数据之间的关系。下面对图形图像处理的各种技术进行讲解。

12.3.1　创建一个简单的图像

使用 GD2 函数库可以实现各种图形图像的处理。创建画布是使用 GD2 函数库来创建图像的第一步，无论创建什么样的图像，首先都需要创建一个画布，其他操作都将在这个画布上完成。在 GD2 函数库中创建画布，可以通过 imagecreate()函数实现。

【例 12.1】 使用 imagecreate()函数创建一个宽 200 像素、高 60 像素的画布，并且设置画布背景颜色 RGB 值为（225，66，159），最后输出一个 gif 格式的图像。实例代码如下：**（实例位置：光盘\TM\sl\12\1）**

```
<?php
$im = imagecreate(200,60);                              //创建一个画布
$white = imagecolorallocate($im, 225,66,159);           //设置画布的背景颜色为粉色
imagegif($im);                                          //输出图像
?>
```

在上面的代码中，主要使用 imagecreate()函数创建一个基于普通调色板的画布，通常支持 256 色，其中 200、60 为图像的宽度和高度，单位为像素（pixel）。

运行结果如图 12.3 所示。



图 12.3　创建一个简单的图像

12.3.2　使用 GD2 函数在照片上添加文字

PHP 中的 GD 库支持中文，但必须要以 UTF-8 格式的参数来进行传递，如果使用 imageString()函数直接绘制中文字符串就会显示乱码，这是因为 GD2 对中文只能接收 UTF-8 编码格式，并且默认使用英文字体，所以要输出中文字符串，必须对中文字符串进行转码，并设置中文字符使用的字体。否则，输出的只能是乱码。



【例 12.2】 使用 imageTTFText()函数将文字“长白山天池”以 TTF（True Type Fonts）字体输出到图像中。（**实例位置：光盘\TM\sl\12\2**）

程序开发步骤如下：

（1）通过 header()函数定义输出图像类型。

（2）通过 imagecreatefromjpeg()函数载入照片。

（3）通过 imagecolorallocate()设置输出字体颜色。

（4）定义输出的中文字符串所使用的字条。

（5）通过 iconv()函数对输出的中文字符串的编码格式进行转换。

（6）通过 imageTTFText()函数向照片中添加文字。

（7）创建图像，并释放资源。

代码如下：

```
<?php
header("content-type:image/jpeg");                              //定义输出为图像类型
$im=imagecreatefromjpeg("images/photo.jpg");                    //载入照片
$textcolor=imagecolorallocate($im,56,73,136);                   //设置字体颜色为蓝色，值为 RGB 颜色值
$fnt="c:/windows/fonts/simhei.ttf";                             //定义字体
$motto=iconv("gb2312","utf-8","长白山天池");                      //定义输出字体串
imageTTFText($im,220,0,480,340,$textcolor,$fnt,$motto);         //写 TTF 文字到图中
imagejpeg($im);                                                 //建立 JPEG 图形
imagedestroy($im);                                              //结束图形，释放内存空间
?>
```

在上面的代码中，主要使用 imageTTFText()函数输出文字到照片中。其中，$im 是指照片，220 是字体的大小，0 是文字的水平方向，480、340 是文字的坐标值，$textcolor 是文字的颜色，$fnt 是字体，$motto 是照片文字。

本实例运行前后的效果如图 12.4 和图 12.5 所示。



图 12.4　照片原图



图 12.5　添加文字后的照片



技巧

应用该方法还可以制作电子相册。

12.3.3　使用图像处理技术生成验证码

验证码功能的实现方法很多，有数字验证码、图形验证码和文字验证码等。在本节中介绍一种使

用图像处理技术生成的验证码。

【例 12.3】 下面介绍使用图像处理技术生成验证码的实现过程。程序的开发步骤如下：**（实例位置：光盘\TM\sl\12\3）**

（1）创建一个 checks.php 文件，在该文件中使用 GD2 函数创建一个 4 位的验证码，并且将生成的验证码保存在 Session 变量中，代码如下：

```
<?php
session_start();                                          //初始化 Session 变量
header("content-type:image/png");                         //设置创建图像的格式
$image_width=70;                                          //设置图像宽度
$image_height=18;                                         //设置图像高度
srand(microtime()*100000);                                //设置随机数的种子
for($i=0;$i<4;$i++){                                      //循环输出一个 4 位的随机数
    $new_number.=dechex(rand(0,15));
}
$_SESSION[check_checks]=$new_number;                      //将获取的随机数验证码写入到 Session 变量中
$num_image=imagecreate($image_width,$image_height);`      //创建一个画布
imagecolorallocate($num_image,255,255,255);               //设置画布的颜色
for($i=0;$i<strlen($_SESSION[check_checks]);$i++){        //循环读取 Session 变量中的验证码
    $font=mt_rand(3,5);                                   //设置随机的字体
    $x=mt_rand(1,8)+$image_width*$i/4;                    //设置随机字符所在位置的 X 坐标
    $y=mt_rand(1,$image_height/4);                        //设置随机字符所在位置的 Y 坐标
    $color=imagecolorallocate($num_image,mt_rand(0,100),mt_rand(0,150),mt_rand(0,200));    //设置字符的颜色
    imagestring($num_image,$font,$x,$y,$_SESSION[check_checks][$i],$color);                //水平输出字符
}
imagepng($num_image);                                     //生成 PNG 格式的图像
imagedestroy($num_image);                                 //释放图像资源
?>
```

在上面的代码中，对验证码进行输出时，每个字符的位置、颜色和字体都是通过随机数来获取的，可以在浏览器中生成各式各样的验证码，还可以防止恶意用户攻击网站系统。

（2）创建一个用户登录的表单，并调用 checks.php 文件，在表单页中输出图像的内容，提交表单信息，使用 if 条件语句判断输入的验证码是否正确。如果用户填写的验证码与随机产生的验证码相等，则提示“用户登录成功!”，代码如下：

```
<?php
session_start();                                          //初始化 Session
if($_POST["Submit"]!=""){
$checks=$_POST["checks"];                                 //获取验证码文本框的值
if($checks==""){                                          //如果验证码的值为空，则弹出提示信息
echo "<script> alert('验证码不能为空');window.location.href='index.php';</script>";
}
//如果用户输入验证码的值与随机生成的验证码的值相等，则弹出登录成功提示
if($checks==$_SESSION[check_checks]){
     echo "<script> alert('用户登录成功!');window.location.href='index.php';</script>";
}else{                                                    //否则弹出验证码不正确的提示信息
     echo "<script> alert('您输入的验证码不正确!');window.location.href='index.php';</script>";
```

```
}
}
?>
```

（3）在 IE 地址栏中输入地址，按 Enter 键，输入用户名和密码，在“验证码”文本框中输入验证码信息，单击“登录”按钮，对验证码的值进行判断，运行结果如图 12.6 所示。



图 12.6　使用图像处理技术生成验证码

12.3.4　使用柱形图统计图书月销售量

柱形图的使用在 Web 网站中非常广泛，它可以直观地显示数据信息，使数据对比和变化趋势一目了然，从而可以更加准确、直观地表达信息和观点。

【例 12.4】 使用 Jpgraph 类库实现柱形图统计图书月销售情况。创建柱形分析图的详细步骤如下：**（实例位置：光盘\TM\sl\12\4）**

（1）使用 include 语句引用 jpgraph.php 文件。

（2）采用柱形图进行统计分析，需要创建 BarPlot 对象，BarPlot 类在 jpgraph_bar.php 中定义，需要使用 include 语句调用该文件。

（3）定义一个 12 个元素的数组，分别表示 12 个月中的图书销量。

（4）创建 Graph 对象，生成一个 600×300 像素大小的画布，设置统计图所在画布的位置以及画布的阴影、淡蓝色背景等。

（5）创建一个矩形的对象 BarPlot，设置其柱形图的颜色，在柱形图上方显示图书销售数据，并格式化数据为整型。

（6）将绘制的柱形图添加到画布中。

（7）添加标题名称和 X 轴坐标，并分别设置其字体。

（8）输出图像。

本实例的完整代码如下：

```
<?php
include ("jpgraph/jpgraph.php");
include ("jpgraph/jpgraph_bar.php");                          //引用柱形图对象所在的文件
$datay=array(160,180,203,289,405,488,489,408,299,166,187,105);   //定义数组
$graph = new Graph(600,300,"auto");                           //创建画布
$graph->SetScale("textlin");
$graph->yaxis->scale->SetGrace(20);
```

```
$graph->SetShadow();                                                   //创建画布阴影
//设置统计图所在画布的位置，左边距 40、右边距 30、上边距 30、下边距 40，单位为像素
$graph->img->SetMargin(40,30,30,40);
$bplot = new BarPlot($datay);                                          //创建一个矩形的对象
$bplot->SetFillColor('orange');                                        //设置柱形图的颜色
$bplot->value->Show();                                                 //设置显示数字
$bplot->value->SetFormat('%d');                                        //在柱形图中显示格式化的图书销量
$graph->Add($bplot);                                                   //将柱形图添加到图像中
$graph->SetMarginColor("lightblue");                                   //设置画布背景色为淡蓝色
$graph->title->Set("《PHP 从入门到精通》2009 年销量统计");              //创建标题
//设置 X 坐标轴文字
$a=array("1 月","2 月","3 月","4 月","5 月","6 月","7 月","8 月","9 月","10 月","11 月","12 月");
$graph->xaxis->SetTickLabels($a);                                      //设置 X 轴
$graph->title->SetFont(FF_SIMSUN);                                     //设置标题的字体
$graph->xaxis->SetFont(FF_SIMSUN);                                     //设置 X 轴的字体
$graph->Stroke();                                                      //输出图像
?>
```

本实例的运行结果如图 12.7 所示。

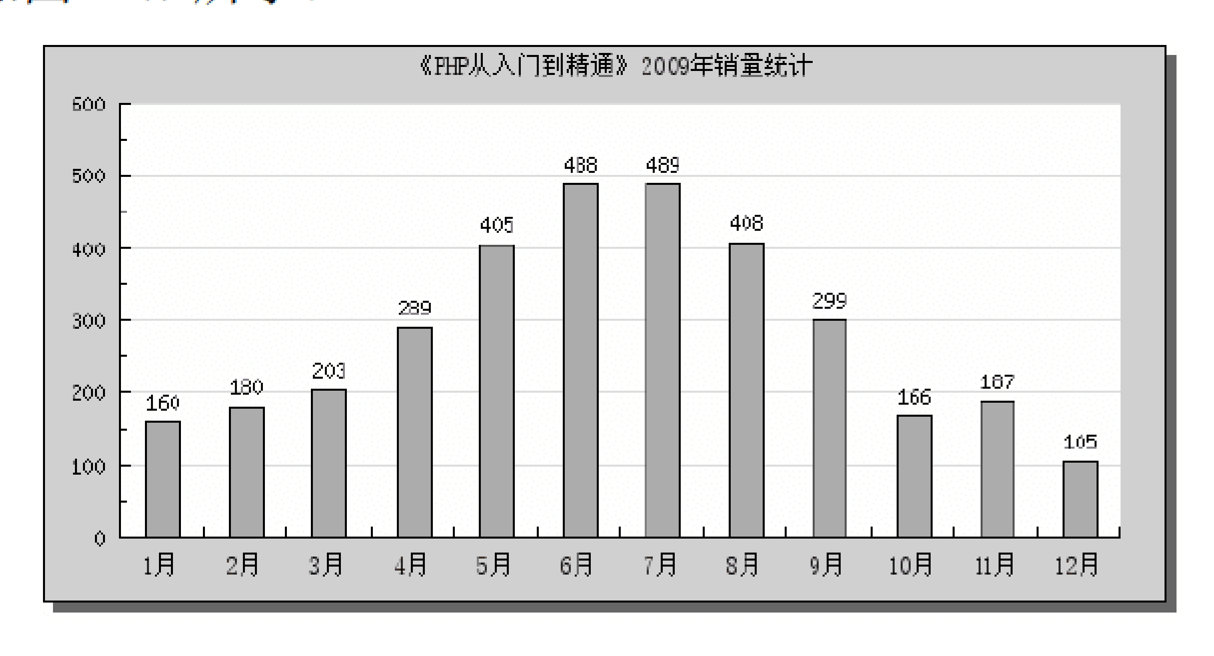


图 12.7　应用柱形图统计图书月销量

12.3.5　使用折线图统计图书月销售额

折线图的使用同样十分广泛，如商品的价格走势、股票在某一时间段的涨跌等，都可以使用折线图来分析。

【例 12.5】 使用 Jpgraph 类库实现折线图统计图书月销售额情况。创建折线分析图的详细步骤如下：**（实例位置：光盘\TM\sl\12\5）**

（1）使用 include 语句引用 jpgraph_line.php 文件。

（2）采用折线图进行统计分析，需要创建 LinePlot 对象，而 LinePlot 类在 jpgraph_line.php 中定义，需要应用 include 语句调用该文件。

（3）定义一个 12 个元素的数组，分别表示 12 个月中的图书月销售额。

（4）创建 Graph 对象，生成一个 600×300 像素大小的画布，设置统计图所在画布的位置，以及画布的阴影、淡蓝色背景等。

（5）创建一个折线图的对象 BarPlot，设置其折线图的颜色。



（6）将绘制的折线图添加到画布中。

（7）添加标题名称和 X 轴坐标，并分别设置其字体。

（8）输出图像。

本实例的完整代码如下：

```
<?php
  include ("jpgraph/jpgraph.php");
  include ("jpgraph/jpgraph_line.php");                                    //引用折线图 LinePlot 类文件
  $datay = array(8320,9360,14956,17028,13060,15376,25428,16216,28548,18632,22724,28460);   //定义数组
  $graph = new Graph(600,300,"auto");                                      //创建画布
  //设置统计图所在画布的位置，左边距 50、右边距 40、上边距 30、下边距 40，单位为像素
  $graph->img->SetMargin(50,40,30,40);
  $graph->img->SetAntiAliasing();                                          //设置折线的平滑状态
  $graph->SetScale("textlin");                                             //设置刻度样式
  $graph->SetShadow();                                                     //创建画布阴影
  $graph->title->Set("2009 年《PHP 从入门到精通》图书月销售额折线图"); //设置标题
  $graph->title->SetFont(FF_SIMSUN,FS_BOLD);                               //设置标题字体
  $graph->SetMarginColor("lightblue");                                     //设置画布的背景颜色为淡蓝色
  $graph->yaxis->title->SetFont(FF_SIMSUN,FS_BOLD);                        //设置 Y 轴标题的字体
  $graph->xaxis->SetPos("min");
  $graph->yaxis->HideZeroLabel();
  $graph->ygrid->SetFill(true,'#EFEFEF@0.5','#BBCCFF@0.5');
  $a=array("1 月","2 月","3 月","4 月","5 月","6 月","7 月","8 月","9 月","10 月","11 月","12 月");//X 轴
  $graph->xaxis->SetTickLabels($a);                                        //设置 X 轴
  $graph->xaxis->SetFont(FF_SIMSUN);                                       //设置 X 坐标轴的字体
  $graph->yscale->SetGrace(20);
  $p1 = new LinePlot($datay);                                              //创建折线图对象
  $p1->mark->SetType(MARK_FILLEDCIRCLE);                                   //设置数据坐标点为圆形标记
  $p1->mark->SetFillColor("red");                                          //设置填充的颜色
  $p1->mark->SetWidth(4);                                                  //设置圆形标记的直径为 4 像素
  $p1->SetColor("blue");                                                   //设置折线颜色为蓝色
  $p1->SetCenter();                                                        //在 X 轴的各坐标点中心位置绘制折线
  $graph->Add($p1);                                                        //在统计图上绘制折线
  $graph->Stroke();                                                        //输出图像
?>
```

本实例的运行结果如图 12.8 所示。

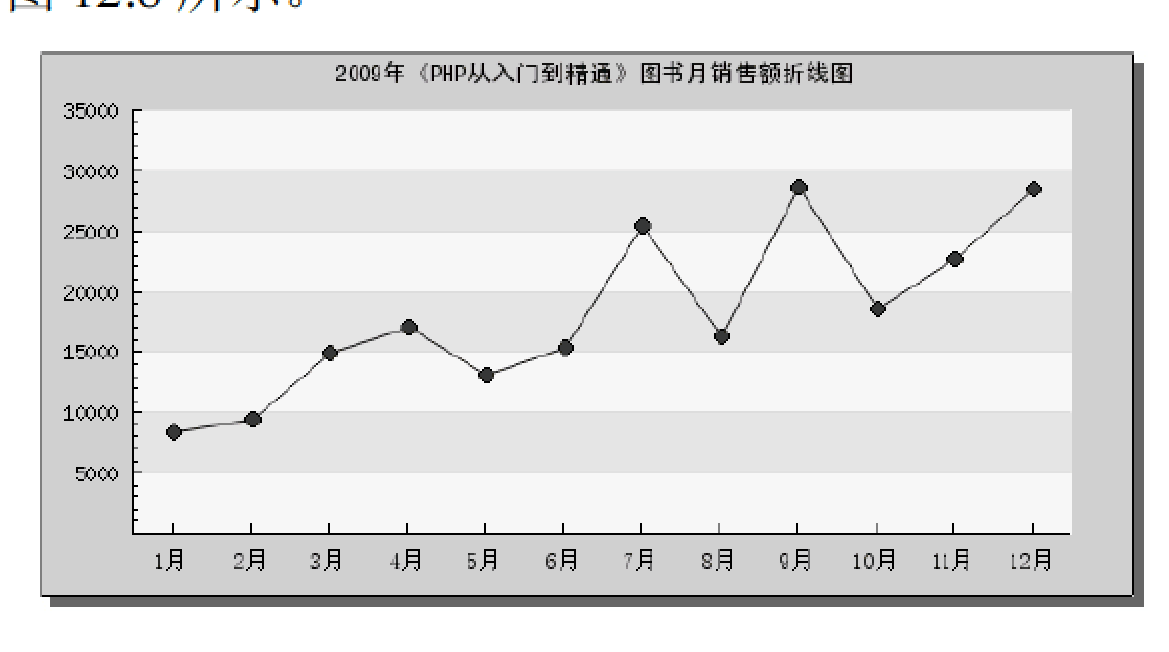


图 12.8　应用折线图统计图书月销售额

12.3.6 使用 3D 饼形图统计各类商品的年销售额比率

饼形图是一种非常实用的数据分析技术，可以清晰地表达出数据之间的关系。在调查某类商品的市场占有率时，最好的显示方式就是使用饼形图，通过饼形图可以直观地看到某类产品的不同品牌在市场中的占有比例。

【例 12.6】 使用 3D 饼形图统计各类商品的年销售额比率。创建 3D 饼形图的详细步骤如下：**（实例位置：光盘\TM\sl\12\6）**

（1）使用 include 语句引用 jpgraph_line.php 文件。

（2）绘制饼形图需要引用 jpgraph_pie.php 文件。

（3）绘制 3D 效果的饼形图需要创建 PiePlot3D 类对象，PiePlot3D 类在 jpgraph_line.php 中定义，需要应用 inlcude 语句调用该文件。

（4）定义一个 6 个元素的数组，分别表示 6 种不同的商品类别。

（5）创建 Graph 对象，生成一个 540×260 像素大小的画布，设置统计图所在画布的位置以及画布的阴影。

（6）设置标题的字体以及图例的字体。

（7）设置饼形图所在画布的位置。

（8）将绘制的 3D 饼形图添加到图像中。

（9）输出图像。

创建 3D 饼形图的程序完整代码如下：

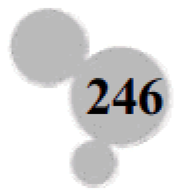
```
<?php
include_once ("jpgraph/jpgraph.php");
include_once ("jpgraph/jpgraph_pie.php");
include_once ("jpgraph/jpgraph_pie3d.php");                 //引用 3D 饼形图 PiePlot3D 对象所在的类文件

$data = array(266036,295621,335851,254256,254254,685425);           //定义数组
$graph = new PieGraph(540,260,'auto');                              //创建画布
$graph->SetShadow();                                                //设置画布阴影

$graph->title->Set("应用 3D 饼形图统计 2009 年商品的年销售额比率");   //创建标题
$graph->title->SetFont(FF_SIMSUN,FS_BOLD);                          //设置标题字体
$graph->legend->SetFont(FF_SIMSUN,FS_NORMAL);                       //设置图例字体

$p1 = new PiePlot3D($data);                                         //创建 3D 饼形图对象
$p1->SetLegends(array("IT 数码","家电通讯","家居日用","服装鞋帽","健康美容","食品烟酒"));
$targ=array("pie3d_csimex1.php?v=1","pie3d_csimex1.php?v=2","pie3d_csimex1.php?v=3",
            "pie3d_csimex1.php?v=4","pie3d_csimex1.php?v=5","pie3d_csimex1.php?v=6");
$alts=array("val=%d","val=%d","val=%d","val=%d","val=%d","val=%d");
$p1->SetCSIMTargets($targ,$alts);

$p1->SetCenter(0.4,0.5);                                            //设置饼形图所在画布的位置
$graph->Add($p1);                                                   //将 3D 饼形图添加到图像中
```



```
$graph->StrokeCSIM();                                       //输出图像到浏览器
?>
```

代码的加粗部分是需要特别注意的地方，这两行代码分别用于设置标题的字体以及图例的字体。本实例的运行结果如图 12.9 所示。

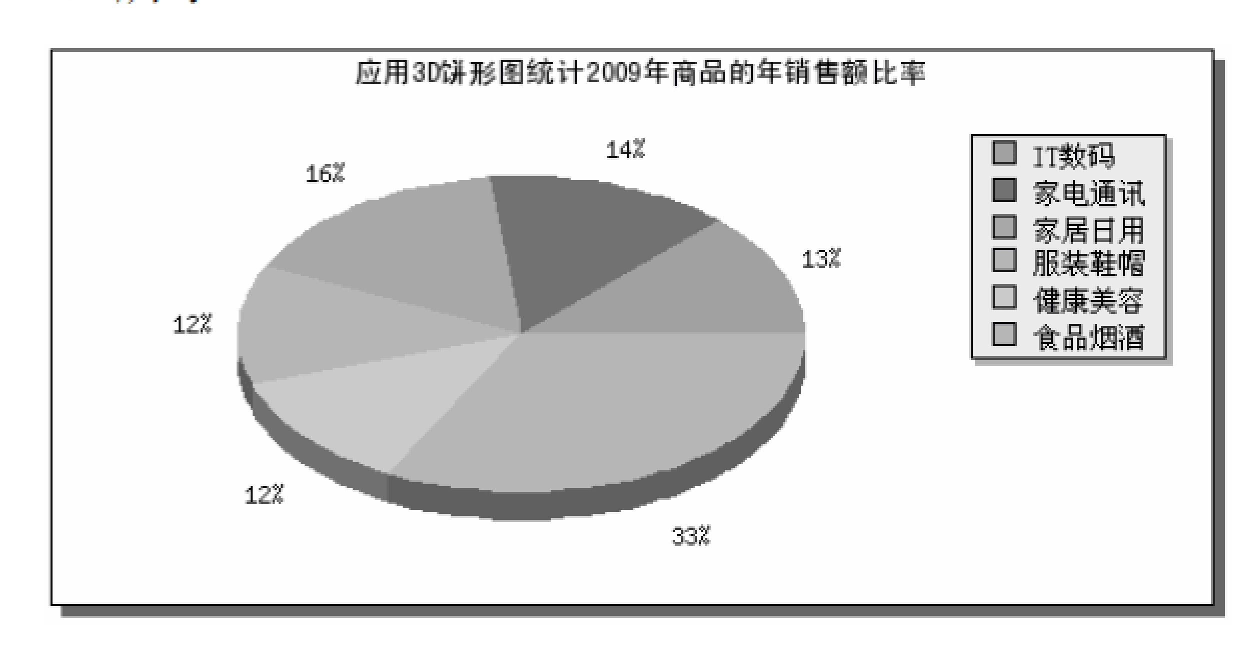


图 12.9　应用 3D 饼形图统计 2009 年各类商品的年销售额比率

12.4　小　　结

本章首先介绍了 GD2 函数库的安装方法，以及应用 GD2 函数创建图像，使读者对 GD2 函数有一个初步的认识。接着介绍了一个专门用于绘制统计图的类库——Jpgraph。通过讲解 Jpgraph 类库的安装、配置到实际的应用过程，指导读者熟练使用该类库，完成更复杂的图形图像的开发。

12.5　练习与实践

1．使用柱形图依次统计 2009 年液晶电视、电冰箱的月销量，要求使用 Jpgraph 类库实现，效果如图 12.10 所示。**（答案位置：光盘\TM\sl\12\7）**

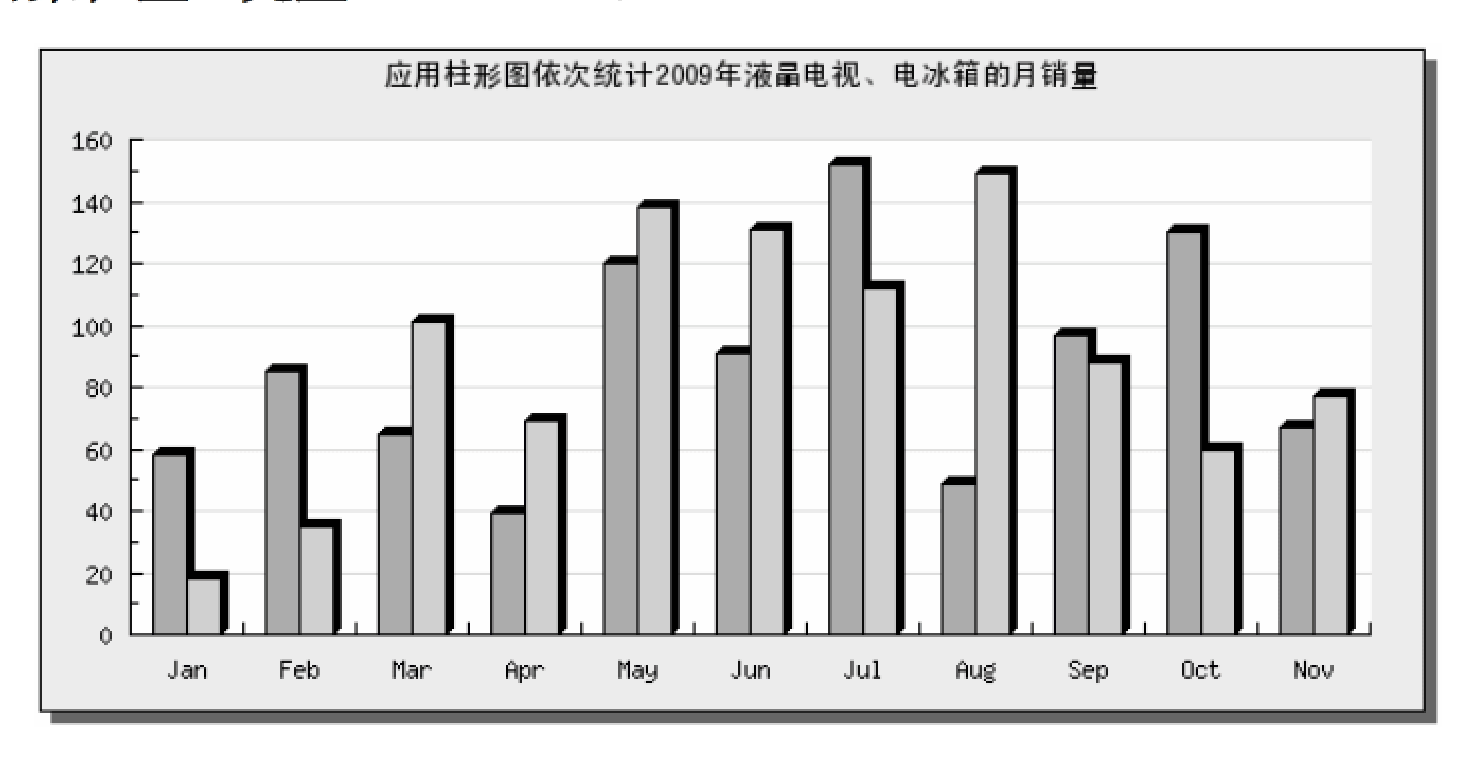


图 12.10　使用柱形图依次统计 2009 年液晶电视、电冰箱的月销量

2．使用折线图统计 2009 年轿车的月销量额，要求使用 Jpgraph 类库实现，效果如图 12.11 所示。**（答案位置：光盘\TM\sl\12\8）**

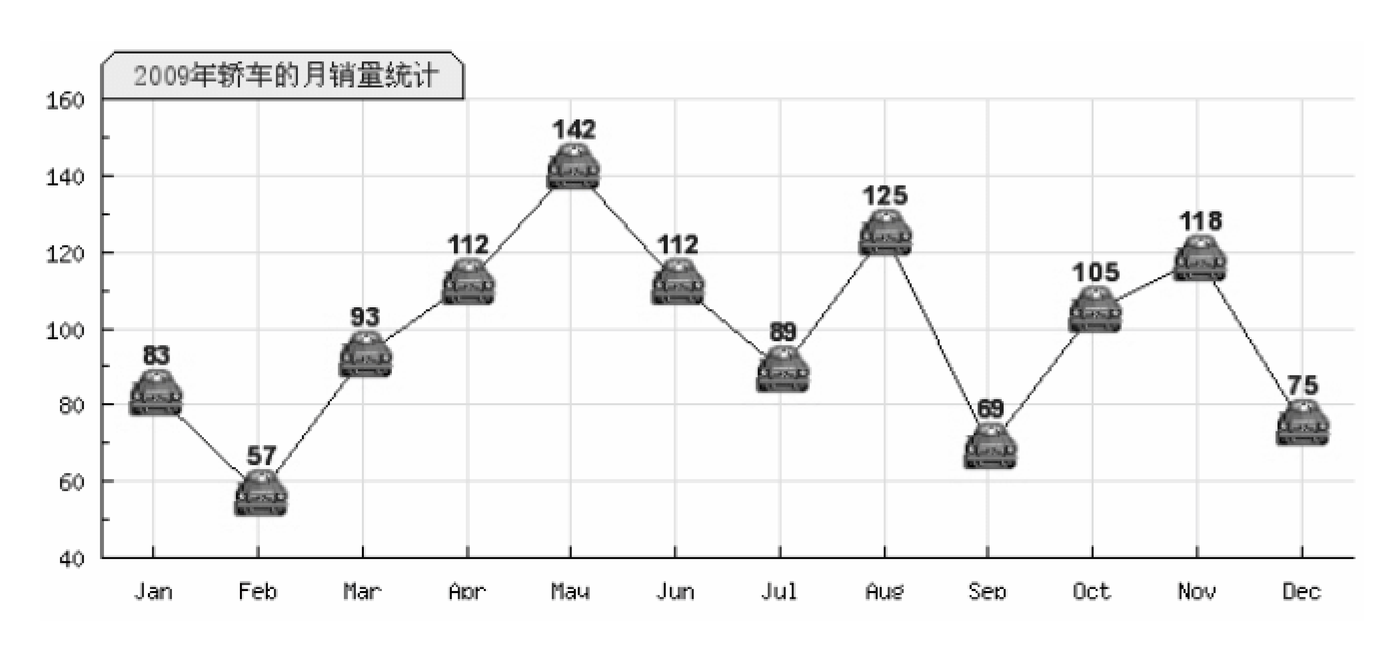


图 12.11　使用折线图统计 2009 年轿车的月销量额

3．使用饼形图统计 2006 年、2007 年、2008 年、2009 年农产品的产量比率，要求使用 Jpgraph 类库实现，效果如图 12.12 所示。**（答案位置：光盘\TM\sl\12\9）**

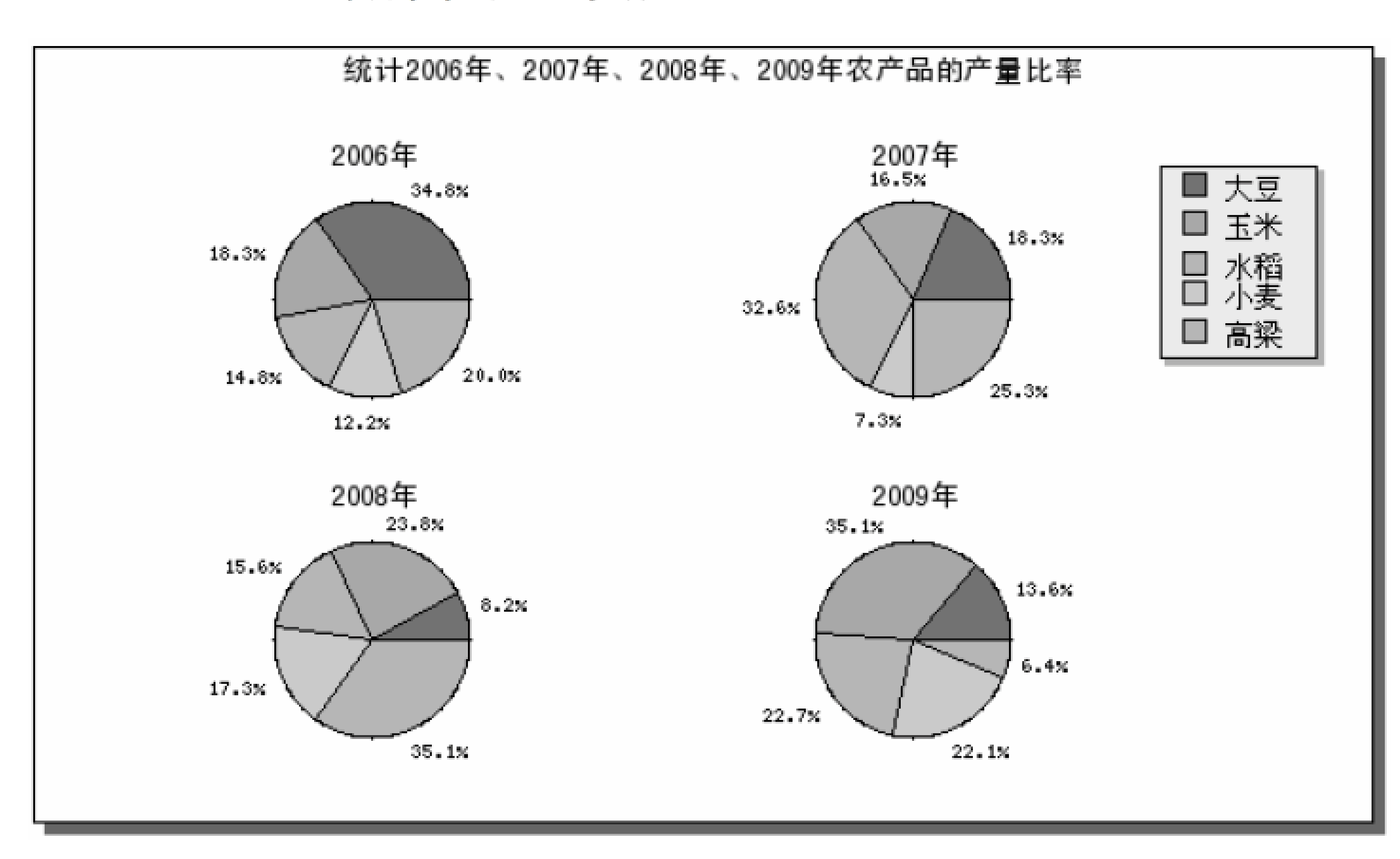


图 12.12　使用饼形图统计 2006 年、2007 年、2008 年、2009 年农产品的产量比率

第13章

文件系统

（视频讲解：34分钟）

文件是用来存取数据的方式之一。相对于数据库来说，文件在使用上更方便、直接。如果数据较少、较简单，使用文件无疑是最合适的方法。PHP能非常好地支持文件上传功能，可以通过配置文件和函数来修改上传功能。

通过阅读本章，您可以：

- 了解如何读、写文件
- 了解如何操作文件
- 掌握目录的处理
- 掌握文件指针的应用
- 掌握锁定文件
- 掌握文件上传

13.1 文 件 处 理

视频讲解：光盘\TM\lx\13\文件处理.exe

文件处理包括读取、关闭、重写等，掌握文件的处理需要读者理清思路，掌握文件处理的关键步骤和常用函数，完全可以运用自如。

例如，访问一个文件需要 3 步：打开文件、读写文件和关闭文件。其他的操作要么是包含在读写文件中（如显示内容、写入内容等），要么与文件自身的属性有关系（如文件遍历、文件改名等）。本节将对常用的文件处理技术进行详细讲解。

13.1.1 打开/关闭文件

打开/关闭文件使用 fopen()函数和 fclose()函数。打开文件应格外认真，一不小心就有可能将文件内容全部删掉。

1. 打开文件

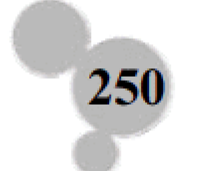
对文件进行操作时首先要打开文件，这是进行数据存取的第一步。在 PHP 中使用 fopen()函数打开文件，fopen()函数的语法如下：

```
resource fopen ( string filename, string mode [, bool use_include_path]);
```

参数 filename 是要打开的包含路径的文件名，可以是相对路径，也可以是绝对路径。如果没有任何前缀则表示打开的是本地文件；参数 mode 是打开文件的方式，可取的值如表 13.1 所示。

表 13.1　fopen()中参数 mode 的取值列表

mode	模 式 名 称	说　　明
r	只读	读模式——进行读取，文件指针位于文件的开头
r+	只读	读写模式——进行读写，文件指针位于文件的开头。在现有文件内容的末尾之前进行写入就会覆盖原有内容
W	只写	写模式——进行写入文件，文件指针指向头文件。如果该文件存在，则所有文件内容被删除；否则函数将创建这个文件
w+	只写	写模式——进行读写，文件指针指向头文件。如果该文件存在，则所有文件的内容被删除；否则函数将创建这个文件
x	谨慎写	写模式打开文件，从文件头开始写。如果文件已经存在，则该文件将不会被打开，函数返回 false，PHP 将产生一个警告
x+	谨慎写	读/写模式打开文件，从文件头开始写。如果文件已经存在，则该文件将不会被打开，函数返回 false，PHP 将产生一个警告
a	追加	追加模式打开文件，文件指针指向尾文件。如果该文件已有内容，则将从文件末尾开始追加；如果该文件不存在，则函数将创建这个文件



续表

mode	模 式 名 称	说　　明
a+	追加	追加模式打开文件，文件指针指向头文件。如果该文件已有内容，则从文件末尾开始追加或者读取；如果该文件不存在，则函数将创建这个文件
b	二进制	二进制模式——用于与其他模式进行连接。如果文件系统能够区分二进制文件和文本文件，可能会使用它。Windows 可以区分；UNIX 则不区分。推荐使用这个选项，便于获得最大程度的可移植性。它是默认模式
t	文本	用于与其他模式的结合。这个模式只是 Windows 下的一个选项

第 3 个参数 use_include_path 是可选的，该参数在配置文件 php.ini 中指定一个路径，如 F:\AppServ\www\mess.php，如果希望服务器在这个路径下打开所指定的文件，可以设置为 1 或 true。

2. 关闭文件

对文件的操作结束后应该关闭这个文件，否则可能引起错误。在 PHP 中使用 fclose()函数关闭文件，该函数的语法如下：

```
bool fclose ( resource handle ) ;
```

该函数将参数 handle 指向的文件关闭，如果成功，返回 true，否则返回 false。其中的文件指针必须是有效的，并且是通过 fopen()函数成功打开的文件。例如：

```
<?php
	$f_open =fopen("../file.txt.","rb");			//打开文件
	…							//对文件进行操作
	fclose($f_open)						//操作完成后关闭文件
?>
```

13.1.2　读写文件

相对打开和关闭文件来说，读写文件更复杂一些。这里主要从读取数据和写入数据两方面讲解。

1. 从文件中读取数据

从文件中读取数据，可以读取一个字符、一行字串或整个文件，还可以读取任意长度的字串。

1）读取整个文件：readfile()、file()和 file_get_contents()

（1）readfile()函数

readfile()函数用于读入一个文件并将其写入到输出缓冲，如果出现错误则返回 false。函数语法如下：

```
int readfile(string filename)
```

使用 readfile()函数，不需要打开/关闭文件，不需要 echo/print 等输出语句，直接写出文件路径即可。

（2）file()函数

file()函数也可以读取整个文件的内容，只是 file()函数将文件内容按行存放到数组中，包括换行符

在内。如果失败则返回 false。函数语法如下：

```
array file(string filename)
```

（3）file_get_contents()函数

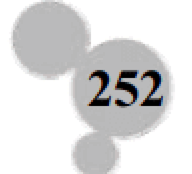
该函数将文件内容（filename）读入一个字符串。如果有 offset 和 maxlen 参数，将在参数 offset 所指定的位置开始读取长度为 maxlen 的内容。如果失败，返回 false。函数语法如下：

```
string file_get_contents(string filename[,int offset[,int maxlen]])
```

该函数适用于二进制对象，是将整个文件的内容读入到一个字符串中的首选方式。

【例 13.1】 本例使用 readfile()函数、file()函数和 file_get_contents()函数分别读取文件 tm.txt 的内容，实例代码如下：**（实例位置：光盘\TM\sl\13\1）**

```
<table width="500" border="1" cellspacing="0" cellpadding="0">
  <tr>
    <td width="253" height="100" align="right" valign="middle" scope="col">使用 readfile()函数读取文件内容：
</td>
    <td width="241" height="100" align="center" valign="middle" scope="col">
    <!--  使用 readfile()函数读取 tm.txt 文件的内容  -->
     <?php readfile('tm.txt'); ?></td>
     <!--  ---------------------------------------------------  -->
  </tr>
  <tr>
    <td height="100" align="right" valign="middle">使用 file()函数读取文件内容：</td>
    <td height="100" align="center" valign="middle">
     <!--  使用 file()函数读取 tm.txt 文件的内容  -->
     <?php
          $f_arr = file('tm.txt');
          foreach($f_arr as $cont){
               echo $cont."<br>";
          }
     ?></td>
     <!--  ---------------------------------------------  -->
  </tr>
  <tr>
     <td width="250" height="25" align="right" valign="middle" scope="col">使用 file_get_contents()函数读取文
件内容：</td>
     <td height="25" align="center" valign="middle" scope="col">
     <!--  使用 file_get_contents()函数读取 tm.txt 文件的内容  -->
     <?php
          $f_chr = file_get_contents('tm.txt');
          echo $f_chr;
     ?></td>
     <!--  -------------------------------------------------------------  -->
  </tr>
</table>
```



运行结果如图 13.1 所示。

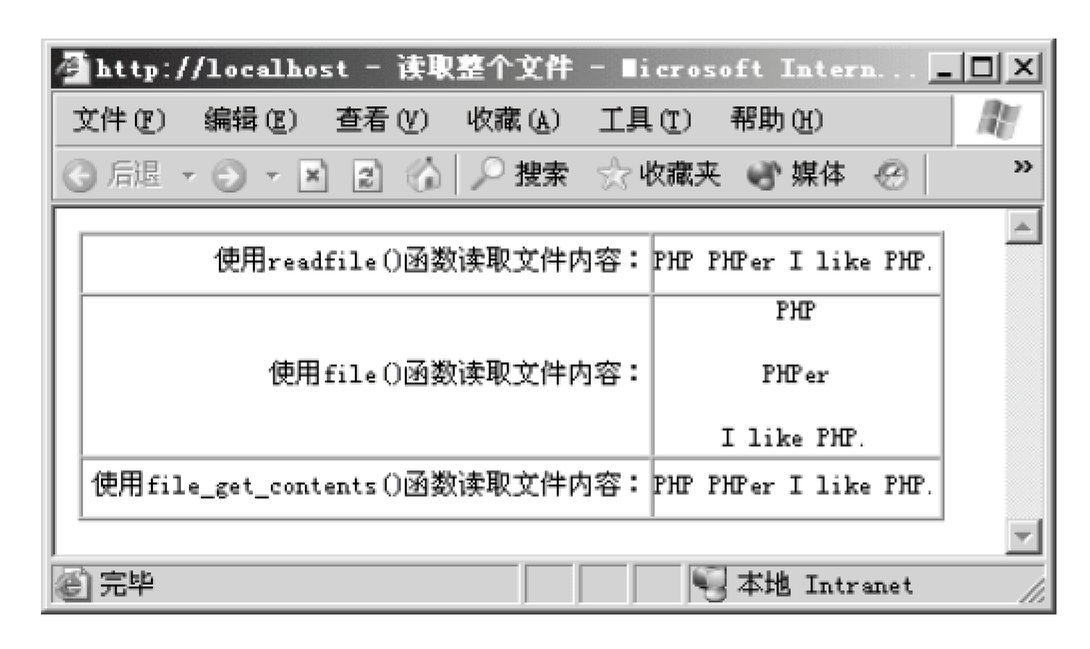


图 13.1　读取整个文件

2）读取一行数据：fgets()和 fgetss()

（1）fgets()函数

fgets()函数用于一次读取一行数据。函数语法如下：

```
string fgets( int handle [, int length] )
```

参数 handle 是被打开的文件，参数 length 是要读取的数据长度。函数能够实现从 handle 指定文件中读取一行并返回长度最大值为 length-1 个字节的字符串。在遇到换行符、EOF 或者读取了 length-1 个字节后停止。如果忽略 length 参数，那么读取数据直到行结束。

（2）fgetss()函数

fgetss()函数是 fgets()函数的变体，用于读取一行数据，同时，fgetss()函数会过滤掉被读取内容中的 html 和 php 标记。函数语法如下：

```
string fgetss ( resource handle [, int length [, string allowable_tags]] )
```

该函数能够从读取的文件中过滤掉任何 html 和 php 标记。可以使用 allowable_tags 参数来控制哪些标记不被过滤掉。

【例 13.2】 使用 fgets()函数与 fgetss()函数分别读取 fun.php 文件并显示出来，观察它们有什么区别。实例代码如下：**（实例位置：光盘\TM\sl\13\2）**

```
<table border="1" cellspacing="0" cellpadding="0">
  <tr>
    <td align="right" valign="middle" scope="col">使用 fgets 函数：</td>
    <td align="center" valign="middle" scope="col">
<!--  使用 fgets 函数读取.php 文件  -->
<?php
     $fopen = fopen('fun.php','rb');
     while(!feof($fopen)){                              //feof()函数测试指针是否到了文件结束的位置
          echo fgets($fopen);                           //输出当前行
     }
     fclose($fopen);
?>
<!--  ----------------------------------------  -->
     </td>
```

```
  </tr>
  <tr>
    <td align="right" valign="middle">使用 fgetss 函数：</td>
    <td align="center" valign="middle">
<!-- 使用 fgetss 函数读取.php 文件 -->
     <?php
          $fopen = fopen('fun.php','rb');
          while(!feof($fopen)){                     //使用 feof()函数测试指针是否到了文件结束的位置
                echo fgetss($fopen);                //输出当前行
          }
     fclose($fopen);
     ?>
     </td>
  </tr>
</table>
```

运行结果如图 13.2 所示。



图 13.2　fgets()函数和 fgetss()函数的区别

3）读取一个字符：fgetc()

在对某一个字符进行查找、替换时，需要有针对性地对某个字符进行读取，在 PHP 中可以使用 fgetc()函数实现此功能。函数语法如下：

```
string fgetc ( resource handle )
```

该函数返回一个字符，该字符从 handle 指向的文件中得到。遇到 EOF 则返回 false。

【例 13.3】 在本例中，使用 fgetc()函数逐个字符读取文件 03.txt 的内容并输出。实例代码如下：**（实例位置：光盘\TM\sl\13\3）**

```
<pre>
<?php
     $fopen = fopen('03.txt','rb');                 //创建文件资源
     while(false !== ($chr = fgetc($fopen))){       //使用 fgetc()函数取得一个字符，判断是否为 false
          echo $chr;                                //如果不是，输出该字符
     }
     fclose($fopen);                                //关闭文件资源
?>
</pre>
```

运行结果如图 13.3 所示。



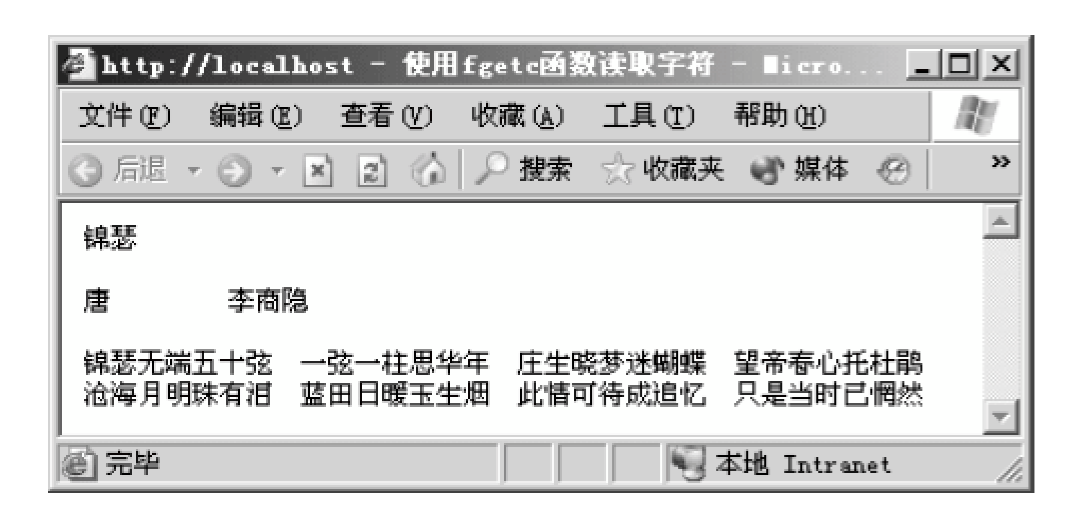


图 13.3　使用 fgetc()函数读取字符

4）读取任意长度的字串：fread()

fread()可以从文件中读取指定长度的数据，函数语法如下：

```
string fread ( int handle, int length )
```

参数 handle 为指向的文件资源，length 是要读取的字节数。当函数读取 length 个字节或到达 EOF 时停止执行。

【例 13.4】 使用 fread()函数读取文件 04.txt 的内容。实例代码如下：（**实例位置：光盘\TM\sl\13\4**）

```
<?php
    $filename = "04.txt";                          //要读取的文件
    $fp = fopen($filename,"rb");                   //打开文件
    echo fread($fp,32);                            //使用 fread()函数读取文件内容的前 32 个字节
    echo "<p>";
    echo fread($fp,filesize($filename));           //输出其余的文件内容
?>
```

运行结果如图 13.4 所示。

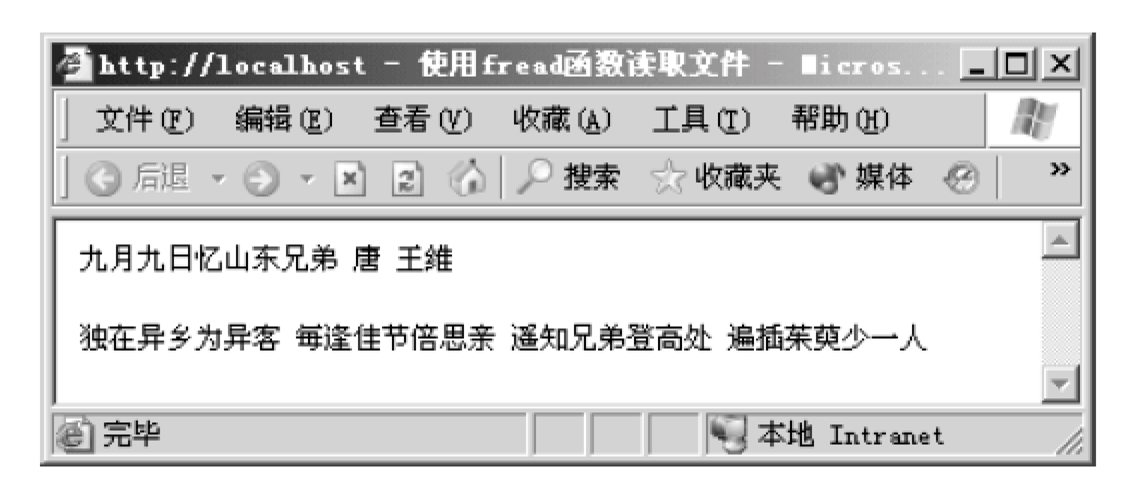


图 13.4　使用 fread()函数读取文件

2．将数据写入文件

写入数据也是 PHP 中常用的文件操作，在 PHP 中使用 fwrite()和 file_put_contents()函数向文件中写入数据。fwrite()函数也称为 fputs()，它们的用法相同。fwrite()函数的语法如下：

```
int fwrite ( resource handle, string string [, int length] )
```

该函数把内容 string 写入文件指针 handle 处。如果指定了长度 length，则写入 length 个字节后停止。如果文件内容长度小于 length，则会输出全部文件内容。

file_put_contents()函数是 PHP 5 新增的函数，其语法为：

```
int file_put_contents ( string filename, string data [, int flags])
```

☑ filename 为写入数据的文件。
☑ data 为要写入的数据。
☑ flags 可以是 FILE_USE_INCLUDE_PATH、FILE_APPEND 或 LOCK_EX，LOCK_EX 为独占锁定，在 13.3.3 节锁定文件中将会介绍。

使用 file_put_contents()函数和依次调用 fopen()、fwrite()、fclose()函数的功能一样。下面通过实例比较一下该函数的优越性。

【例 13.5】 本实例首先使用 fwrite()函数向 05.txt 文件写入数据，再使用 file_put_contents()函数写入数据。实例代码如下：**（实例位置：光盘\TM\sl\13\5）**

```
<?php
	$filepath = "05.txt";
	$str = "此情可待成追忆　只是当时已惘然<br>";
	echo "用 fwrite 函数写入文件：";
	$fopen = fopen($filepath,'wb') or die('文件不存在');
	fwrite($fopen,$str);
	fclose($fopen);
	readfile($filepath);
	echo "<p>用 file_put_contents 函数写入文件：";
	file_put_contents($filepath,$str);
	readfile($filepath);
?>
```

运行结果如图 13.5 所示。

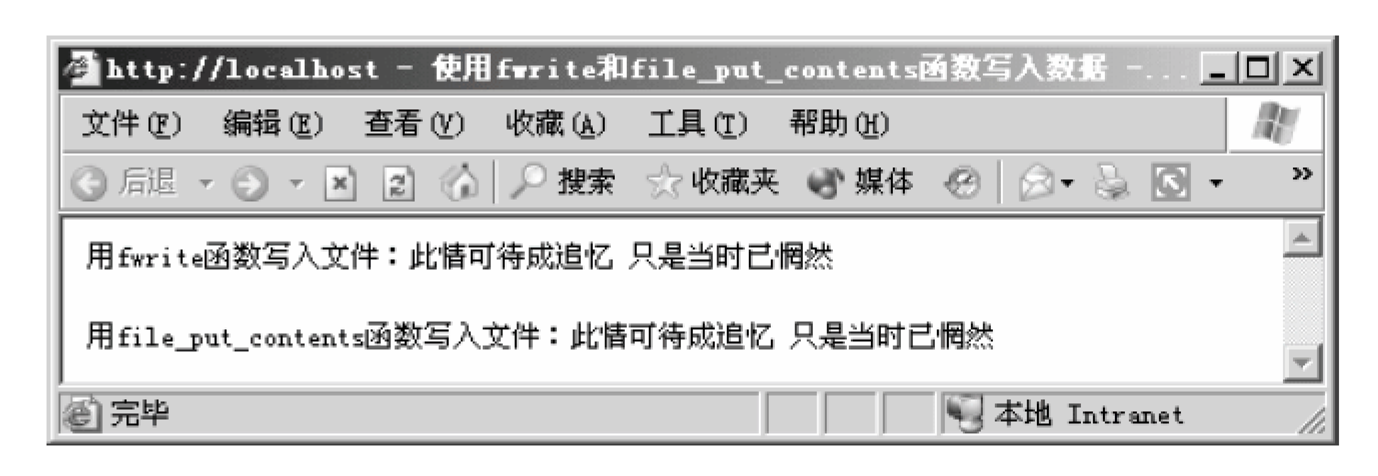


图 13.5　使用 fwrite()和 file_put_contents()函数写入数据

13.1.3　操作文件

除了可以对文件内容进行读写，对文件本身同样也可以进行操作，如复制、重命名、查看修改日期等。PHP 内置了大量的文件操作函数，常用的文件函数如表 13.2 所示。

表 13.2　常用的文件操作函数

函 数 原 型	函 数 说 明	举　例
bool copy(string path1, string path2)	将文件从 path1 复制到 path2。如果成功，返回 true，失败则返回 false	copy('tm.txt','../tm.txt')
bool rename(string filename1,string filename2)	把 name1 重命名为 name2	rename('1.txt','tm.txt')



续表

函 数 原 型	函 数 说 明	举 例
bool unlink(string filename)	删除文件，成功返回 true，失败则返回 false	unlink('./tm.txt')
int fileatime(string filename)	返回文件最后一次被访问的时间，时间以 UNIX 时间戳的方式返回	fileatime('1.txt')
int filemtime(string filename)	返回文件最后一次被修改的时间，时间以 UNIX 时间戳的方式返回	date('Y-m-d H:i:s', filemtime('1.txt'))
int filesize(string filename)	取得文件 filename 的大小（bytes）	filesize('1.txt')

array pathinfo(string name [, int options])	返回一个数组，包含文件 name 的路径信息。有 dirname、basename 和 extension。可以通过 option 设置要返回的信息，有 PATHINFO_DIRNAME、PATHINFO_BASENAME 和PATHINFO_EXTENSION。默认为返回全部	$arr = pathinfo('/tm/sl/12/5/1.txt');
		foreach($arr as $method => $value){
		echo $method.": ".$value."
		";
		}

string realpath (string filename)	返回文件 filename 的绝对路径。如 c:\tmp\…\1.txt	realpath('1.txt')

array stat (string filename)	返回一个数组，包括文件的相关信息，如上面提到的文件大小、最后修改时间等	$arr = stat('1.txt');
		foreach($arr as $method => $value){
		echo $method.": ".$value."
		";
		}

说明

在读写文件时，除了 file()、readfile()等少数几个函数外，其他操作必须要先使用 fopen()函数打开文件，最后用 fclose()函数关闭文件。文件的信息函数（如 filesize、filemtime 等）则都不需要打开文件，只要文件存在即可。

13.2 目 录 处 理

视频讲解：光盘\TM\lx\13\目录处理.exe

目录是一种特殊的文件。要浏览目录下的文件，首先要打开目录，浏览完毕后，同样要关闭目录。目录处理包括打开目录、浏览目录和关闭目录。

13.2.1 打开/关闭目录

打开/关闭目录和打开/关闭文件类似，但打开的文件如果不存在，就自动创建一个新文件，而打开的文件路径如果不正确，则一定会报错。

1. 打开目录

PHP 使用 opendir()函数来打开目录，函数语法如下：

```
resource opendir ( string path)
```

函数 opendir()的参数 path 是一个合法的目录路径，成功执行后返回目录的指针；如果 path 不是一个合法的目录或者因为权限或文件系统错误而不能打开目录，则返回 false 并产生一个 E_WARNING 级别的错误信息。可以在 opendir()前面加上“@”符号来抑制错误信息的输出。

2. 关闭目录

关闭目录使用 closedir()函数，函数语法如下：

```
void closedir ( resource handle )
```

参数 handle 为使用 opendir()函数打开的一个目录指针。
下面为打开和关闭目录的流程代码：

```
<?php
$path = "D:\\AppServ\\www\\tm\\sl\\12" ;
if (is_dir($path)){                          //检测是否是一个目录
      if ($dire = opendir($path))            //判断打开目录是否成功
            echo $dire;                      //输出目录指针
}else{
      echo '路径错误';
      exit();
}
…                                            //其他操作
closedir($dire);                             //关闭目录
?>
```

is_dir()函数判断当前路径是否为一个合法的目录。如果合法，返回 true，否则返回 false。

13.2.2 浏览目录

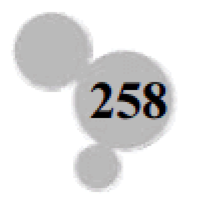
在 PHP 中浏览目录中的文件使用的是 scandir()函数，函数语法如下：

```
array scandir ( string directory [, int sorting_order ])
```

该函数返回一个数组，包含 directory 中的所有文件和目录。参数 sorting_order 指定排序顺序，默认按字母升序排序，如果添加了该参数，则变为降序排序。

【例 13.6】 本例将查看 F:\AppServ\www\tm\13 目录下的所有文件。实例代码如下：（**实例位置：光盘\TM\ sl\13\6**）

```
<?php
      $path = 'F:\AppServ\www\tm\13';                  //要浏览的目录
```



```
        if(is_dir($path)){                                   //判断文件名是否为目录
                $dir = scandir($path);                       //使用 scandir 函数取得所有文件及目录
                foreach($dir as $value){                     //使用 foreach 循环
                        echo $value."<br>";                  //循环输出文件及目录名称
                }
        }else{
                echo "目录路径错误！";
        }
?>
```

运行结果如图 13.6 所示。



图 13.6　浏览目录

13.2.3　操作目录

目录是特殊的文件，也就是说，对文件的操作处理函数（如重命名）多数同样适用于目录。但还有一些特殊的函数只是针对目录的，表 13.3 列举了一些常用的目录操作函数。

表 13.3　常用的目录操作函数

函 数 原 型	函 数 说 明	举　　例
bool mkdir (string pathname)	新建一个指定的目录	mkdir('temp');
bool rmdir (string dirname)	删除所指定的目录，该目录必须是空的	rmdir('tmp')
string getcwd (void)	取得当前工作的目录	getcwd()
bool chdir (string directory)	改变当前目录为 directory	echo getcwd() . " "; chdir('../'); echo getcwd() . " ";
float disk_free_space (string directory)	返回目录中的可用空间（bytes）。被检查的文件必须通过服务器的文件系统访问	disk_free_space('d:\\appserv');
float disk_total_space(string directory)	返回目录的总空间大小（bytes）	disk_total_space('d:\\appserv');

string readdir (resource handle)	返回目录中下一个文件的文件名(使用此函数时，目录必须是使用 opendir()函数打开的）。在 PHP 5 之前，都是使用这个函数来浏览目录的	while(false!==($path=readdir($handle))){
		echo $path;
		}

void rewinddir (resource handle)	将指定的目录重新指定到目录的开头	rewinddir($handle)

13.3 文件处理的高级应用

视频讲解：光盘\TM\lx\13\高级应用.exe

在 PHP 中，除了可以对文件进行基本的读写操作外，还可以对文件指针进行查找、定位，对正在读取的文件进行锁定等，本节将进一步学习文件处理的高级技术。

13.3.1 远程文件的访问

PHP 支持 URL 格式的文件调用，只要在 php.ini 中配置一下即可。在 PHP 中找到 allow_url_fopen，将该选项设为 ON。重启服务器后即可使用 HTTP 或 FTP 的 URL 格式。如：

```
fopen('http://127.0.0.1/tm/sl/index.php','rb');
```

13.3.2 文件指针

PHP 可以实现文件指针的定位及查询，从而实现所需信息的快速查询。文件指针函数有 rewind()、fseek()、feof()和 ftell()。

1. rewind()函数

该函数将文件 handle 的指针设为文件流的开头，语法如下：

```
bool rewind ( resource handle )
```

注意

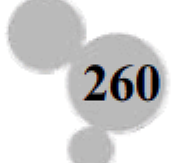
如果将文件以附加（“a”）模式打开，写入文件的任何数据总是会被附加在后面，不论文件指针的位置在何处。

2. fseek()函数

fseek()函数实现文件指针的定位，语法如下：

```
int fseek ( resource handle, int offset [, int whence] )
```

- ☑ handle 参数为要打开的文件。
- ☑ offset 为指针位置或相对 whence 参数的偏移量，可以是负值。
- ☑ whence 的值包括以下 3 种：
  - ➢ SEEK_SET，位置等于 offset 字节。
  - ➢ SEEK_CUR，位置等于当前位置加上 offset 字节。



➢ SEEK_END，位置等于文件尾加上 offset 字节。

如果忽略 whence 参数，系统默认为 SEEK_SET。

3．feof()函数

该函数判断文件指针是否在文件尾，语法如下：

```
bool feof ( resource handle )
```

如果文件指针到了文件结束的位置，就返回 true，否则返回 false。

4．ftell()函数

ftell()函数返回当前指针的位置，语法如下：

```
int ftell ( resource handle )
```

【例 13.7】 下面使用 4 个指针函数来输出文件 07.txt 中的内容，实例代码如下：（**实例位置：光盘\TM\sl\13\7**）

```
<?php
$filename = "07.txt";                                           //指定文件路径及文件名
if(is_file($filename)){                                         //判断文件是否存在
    echo "文件总字节数：".filesize($filename)."<br>";              //输出总字节数
    $fopen = fopen($filename,'rb');                             //打开文件
    echo "初始指针位置是：".ftell($fopen)."<br>";                  //输出指针位置
    fseek($fopen,33);                                           //移动指针
    echo "使用 fseek()函数后指针位置：".ftell($fopen)."<br>";      //输出移动后的指针位置
    echo "输出当前指针后面的内容：".fgets($fopen)."<br>";          //输出从当前指针到行尾的内容
    if(feof($fopen))                                            //判断指针是否指向文件末尾
        echo "当前指针指向文件末尾：".ftell($fopen)."<br>";        //如果指向了文件尾，则输出指针位置
    rewind($fopen);                                             //使用 rewind()函数
    echo "使用 rewind()函数后指针的位置：".ftell($fopen)."<br>";   //查看使用 rewind()函数后指针的位置
    echo "输出前 33 个字节的内容：".fgets($fopen,33);              //输出前 33 个字节的内容
    fclose($fopen);                                             //关闭文件
}else{
    echo "文件不存在";
}
?>
```

运行结果如图 13.7 所示。

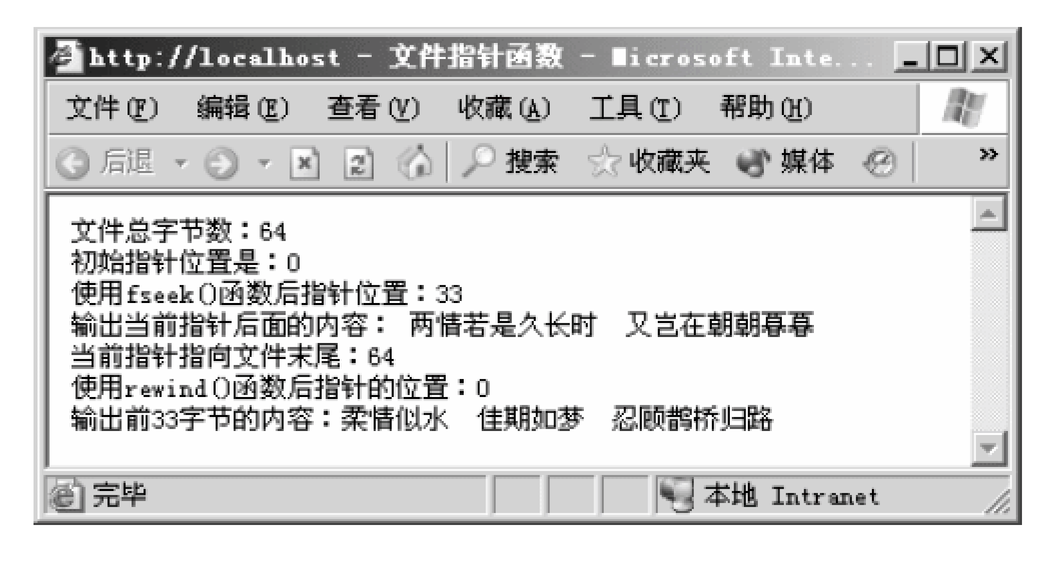


图 13.7　文件指针函数

13.3.3 锁定文件

在向一个文本文件写入内容时，需要先锁定该文件，以防止其他用户同时修改此文件内容。在 PHP 中锁定文件的函数为 flock()，语法如下：

```
bool flock ( int handle, int operation)
```

参数 handle 为一个已经打开的文件指针，operation 的参数如表 13.4 所示。

表 13.4 operation 的参数值

参 数 值	说 明
LOCK_SH	取得共享锁定（读取程序）
LOCK_EX	取得独占锁定（写入程序）
LOCK_UN	释放锁定
LOCK_NB	防止 flock()在锁定时堵塞

【例 13.8】 本例使用 flock()函数锁定文件，然后再写入数据，最后解除锁定，关闭文件。实例代码如下：**（实例位置：光盘\TM\sl\13\8）**

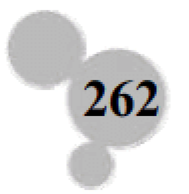
```
<?php
    $filename = '08.txt';                    //声明要打开的文件名称
    $fd = fopen($filename,'w');              //以 w 形式打开文件
    flock($fd, LOCK_EX);                     //锁定文件（独占共享）
    fwrite($fd, "hightman1");                //向文件中写入数据
    flock($fd, LOCK_UN);                     //解除锁定
    fclose($fd);                             //关闭文件指针
    readfile($filename);                     //输出文件内容
?>
```

在向文件写入数据时，使用 w 或 w+选项来打开文件。这时如果使用了 LOCK_EX，则同一时间访问此文件的其他用户无法得到文件的大小，也不能进行写操作。

13.4 文 件 上 传

视频讲解：光盘\TM\lx\13\文件上传.exe

文件上传可以通过 HTTP 协议来实现。要使用文件上传功能，首先要在 php.ini 配置文件中对上传做一些设置，然后了解预定义变量$_FILES，通过$_FILES 的值对上传文件做一些限制和判断，最后使用 move_uploaded_file()函数实现上传。



13.4.1　配置 php.ini 文件

要想顺利地实现上传功能，首先要在 php.ini 中开启文件上传，并对其中的一些参数作出合理的设置。找到 File Uploads 项，可以看到下面有 3 个属性值，表示含义如下。

☑ file_uploads：如果值是 on，说明服务器支持文件上传；如果为 off，则不支持。

☑ upload_tmp_dir：上传文件临时目录。在文件被成功上传之前，文件首先存放到服务器端的临时目录中。如果想要指定位置，可在这里设置。否则使用系统默认目录即可。

☑ upload_max_filesize：服务器允许上传的文件的最大值，以 MB 为单位。系统默认为 2MB，用户可以自行设置。

除了 File Uploads 项，还有几个属性也会影响到上传文件的功能。

☑ max_execution_time：PHP 中一个指令所能执行的最大时间，单位是秒。

☑ memory_limit：PHP 中一个指令所分配的内存空间，单位是 MB。



说明

如果使用集成化的安装包来配置 PHP 的开发环境，上述介绍的这些配置信息默认已经配置好了。

注意

如果要上传超大的文件，需要对 php.ini 进行修改。包括 upload_max_filesize 的最大值，max_execution_time 一个指令所能执行的最大时间和 memory_limit 一个指令所分配的内存空间。

13.4.2　预定义变量$_FILES

$_FILES 变量存储的是上传文件的相关信息，这些信息对于上传功能有很大的作用。该变量是一个二维数组。保存的信息如表 13.5 所示。

表 13.5　预定义变量$_FILES 元素

元　素　名	说　　明
$_FILES[filename][name]	存储了上传文件的文件名。如 exam.txt、myDream.jpg 等
$_FILES[filename][size]	存储了文件大小。单位为字节
$_FILES[filename][tmp_name]	文件上传时，首先在临时目录中被保存成一个临时文件。该变量为临时文件名
$_FILES[filename][type]	上传文件的类型
$_FILES[filename][error]	存储了上传文件的结果。如果返回 0，说明文件上传成功

【例 13.9】 本例创建一个上传文件域，通过$_FILES 变量输出上传文件的资料。实例代码如下：**（实例位置：光盘\TM\sl\13\9）**

```
<table width="500" border="0" cellspacing="0" cellpadding="0">
<!--  上传文件的 form 表单，必须有 enctype 属性  -->
<form action="" method="post" enctype="multipart/form-data">
  <tr>
    <td width="150" height="30" align="right" valign="middle">请选择上传文件：</td>
    <!--  上传文件域，type 类型为 file  -->
    <td width="250"><input type="file" name="upfile"/></td>
    <!--  提交按钮  -->
    <td width="100"><input type="submit" name="submit" value="上传" /></td>
  </tr>
</form>
</table>
<?php
<!--  处理表单返回结果  -->
    if(!empty($_FILES)){                                  //判断变量$_FILES 是否为空
        foreach($_FILES['upfile'] as $name => $value)     //使用 foreach 循环输出上传文件信息的名称和值
            echo $name.' = '.$value.'<br>';
    }
?>
```

运行结果如图 13.8 所示。

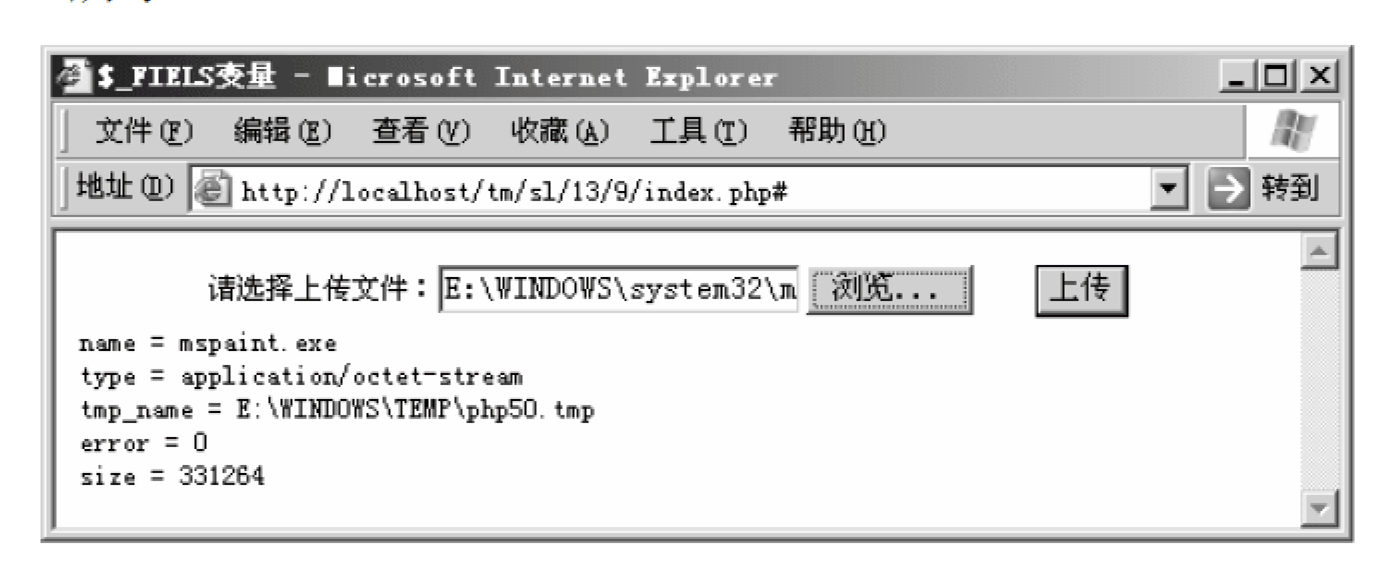


图 13.8　$_FILES 预定义变量

13.4.3　文件上传函数

PHP 中使用 move_uploaded_file()函数上传文件。该函数的语法如下：

```
bool move_uploaded_file ( string filename, string destination )
```

move_uploaded_file()函数将上传文件存储到指定的位置。如果成功，则返回 true，否则返回 false。参数 filename 是上传文件的临时文件名，即$_FILES[tmp_name]；参数 destination 是上传后保存的新的路径和名称。

【例 13.10】 本例创建一个上传表单，允许上传 150KB 以下的文件。实例代码如下：（**实例位置：光盘\TM\sl\13\10**）

```
<!--  上传表单，有一个上传文件域  -->
<form action="" method="post" enctype="multipart/form-data" name="form">
    <input name="up_file" type="file" />
```



```
        <input type="submit" name="submit" value="上传" />
</form>
<!--  ---------------------------------------   -->
<?php
/*   判断是否有上传文件   */
        if(!empty($_FILES[up_file][name])){
/*   将文件信息赋给变量$fileinfo   */
                $fileinfo = $_FILES[up_file];
/*   判断文件大小   */
                if($fileinfo['size'] < 1000000 && $fileinfo['size'] > 0){
/*   上传文件   */
                        move_uploaded_file($fileinfo['tmp_name'],$fileinfo['name']);
                        echo '上传成功';
                }else{
                        echo '文件太大或未知';
                }
        }
?>
```

运行结果如图 13.9 所示。



图 13.9　单文件上传

注意

使用 move_uploaded_file()函数上传文件时，在创建 form 表单时，必须设置 form 表单的 enctype="multipart/form-data"属性。

13.4.4　多文件上传

PHP 支持同时上传多个文件，只需要在表单中对文件上传域使用数组命名即可。

【例 13.11】 本实例有 4 个文件上传域，文件域的名字为 u_file[]，提交后上传的文件信息都被保存到$_FILES[u_file]中，生成多维数组。读取数组信息，并上传文件。实例代码如下：（**实例位置：光盘\TM\sl\13\11**）

```
请选择要上传的文件
<!--   上传文件表单   -->
<form action="" method="post" enctype="multipart/form-data">
<table id="up_table" border="1" bgcolor="f0f0f0" >
```

```
        <tbody id="auto">
            <tr id="show" >
              <td>上传文件 </td>
                <td><input name="u_file[]" type="file"></td>
                </tr>
            <tr>
              <td>上传文件 </td>
                <td><input name="u_file[]" type="file"></td>
                </tr></tbody>
              <tr><td colspan="4"><input type="submit" value="上传" /></td></tr> </table> </form>
<?php
<!--  判断变量$_FILES 是否为空   -->
if(!empty($_FILES[u_file][name])){
      $file_name = $_FILES[u_file][name];                          //将上传文件名另存为数组
      $file_tmp_name = $_FILES[u_file][tmp_name];                  //将上传的临时文件名存为数组
      for($i = 0; $i < count($file_name); $i++){                   //循环上传文件
       if($file_name[$i] != ''){                                   //判断上传文件名是否为空
            move_uploaded_file($file_tmp_name[$i],$i.$file_name[$i]);
            echo '文件'.$file_name[$i].'上传成功。更名为'.$i.$file_name[$i].'<br>';

      }
}
?>
```

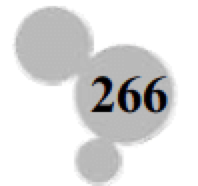
运行结果如图 13.10 所示。



图 13.10　多文件上传

13.5　小　　结

本章首先介绍对文件的基本操作，然后学习目录的基本操作，接下来学习文件的高级处理技术，



最后又学习 PHP 的文件上传技术，这是一个网站必不可少的组成部分。希望读者能够深入理解本章的重点知识，牢固掌握常用函数。

13.6　练习与实践

1．通过文本文件统计页面访问量。**（答案位置：光盘\TM\sl\13\12）**
2．控制上传文件大小。**（答案位置：光盘\TM\sl\13\13）**

第14章

面向对象

（视频讲解：29分钟）

面向对象是一种计算机编程架构，比面向过程编程具有更强的灵活性和扩展性。面向对象编程也是一个程序员发展的“分水岭”，很多的初学者和略有成就的开发者，就是因为无法理解“面向对象”而放弃。这里想提醒一下初学者：要想在编程这条路上走得比别人远，就一定要掌握面向对象编程技术。

通过阅读本章，您可以：

- 了解面向对象的概念
- 了解 PHP 中的面向对象
- 了解类的定义及实例化
- 掌握声明类成员
- 了解继承和多态的实现
- 了解 PHP 对象的高级应用
- 了解抽象类的实现
- 掌握接口的使用
- 了解魔术方法
- 熟悉面向对象的基本应用

14.1 面向对象的基本概念

视频讲解：光盘\TM\lx\14\面向对象的基本概念.exe

这里所指的面向对象，准确地说应该叫做面向对象编程（OOP），是面向对象的一部分。面向对象包括 3 个部分：面向对象分析（Object Oriented Analysis，OOA）、面向对象设计（Object Oriented Design，OOD），以及面向对象编程（Object Oriented Programming，OOP）。面向对象编程的两个重点概念是类和对象。

14.1.1 类

世间万物都具有其自身的属性和方法，通过这些属性和方法可以将不同物质区分开来。例如，人具有身高、体重和肤色等属性，还可以进行吃饭、学习、走路等能动活动，这些活动可以说是人具有的功能。可以把人看作程序中的一个类，那么人的身高可以看作类中的属性，走路可以看作类中的方法。也就是说，类是属性和方法的集合，是面向对象编程方式的核心和基础，通过类可以将零散的用于实现某项功能的代码进行有效管理。例如，创建一个运动类，包括 5 个属性：姓名、身高、体重、年龄和性别，定义 4 个方法：踢足球、打篮球、举重和跳高，如图 14.1 所示。

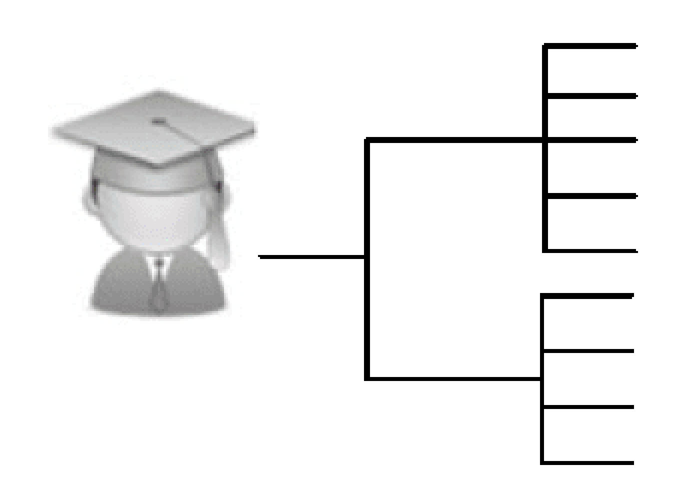


图 14.1 运动类

14.1.2 对象

类只是具备某项功能的抽象模型，实际应用中还需要对类进行实例化，这样就引入了对象的概念。对象是类进行实例化后的产物，是一个实体。仍然以人为例，“黄种人是人”这句话没有错误，但反过来说“人是黄种人”这句话一定是错误的。因为除了有黄种人，还有黑人、白人等。那么“黄种人”就是“人”这个类的一个实例对象。可以这样理解对象和类的关系：对象实际上就是“有血有肉的、能摸得到看得到的”一个类。

这里实例化 14.1.1 节中创建的运动类，调用运动类中的打篮球方法，判断提交的实例对象是否符合打篮球的条件，如图 14.2 所示。



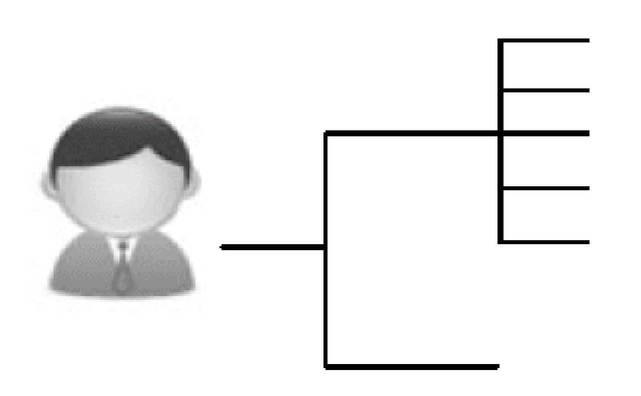


图 14.2　实例化对象

这里根据实例化对象，调用打篮球方法，并向其中传递参数（明日，185 公分，80 公斤，20 周岁，男），在打篮球方法中判断这个对象是否符合打篮球的条件。

14.1.3　面向对象编程的三大特点

面向对象编程的三大特点就是封装性、继承性和多态性。

1. 封装性

封装性，也可以称为信息隐藏。就是将一个类的使用和实现分开，只保留有限的接口（方法）与外部联系。对于用到该类的开发人员，只要知道这个类该如何使用即可，而不用去关心这个类是如何实现的。这样做可以让开发人员更多地把精力集中起来专注别的事情，同时也避免了程序之间的相互依赖而带来不便。

2. 继承性

继承性就是派生类（子类）自动继承一个或多个基类（父类）中的属性与方法，并可以重写或添加新的属性或方法。继承这个特性简化了对象和类的创建，增加了代码的可重用性。继承分单继承和多继承，PHP 所支持的是单继承，也就是说，一个子类有且只有一个父类。

3. 多态性

多态性是指同一个类的不同对象，使用同一个方法可以获得不同的结果，这种技术称为多态性。多态性增强了软件的灵活性和重用性。

14.2　PHP 与对象

视频讲解：光盘\TM\lx\14\PHP 与对象.exe

14.2.1　类的定义

和很多面向对象的语言一样，PHP 也是通过 class 关键字加类名来定义类的。类的格式如下：

```
<?php
class SportObject{                    //定义运动类
        //…
}
?>
```

上述两个大括号中间的部分是类的全部内容，如上述 SportObject 就是一个最简单的类。SportObject 类仅有一个类的骨架，什么功能都没有实现，但这并不影响它的存在。

注意

一个类，即一对大括号之间的全部内容都要在一段代码段中，即一个“<?php … ?>”之间不能分割成多块，例如：

```
<?php
class SportObject{                    //定义运动类
        //…
?>
<?php
        //…
}
?>
```

这种格式是不允许的。

14.2.2 成员方法

类中的函数被称为成员方法。函数和成员方法唯一的区别就是，函数实现的是某个独立的功能，而成员方法是实现类中的一个行为，是类的一部分。

下面就创建在图 14.1 中编写的运动类，并添加成员方法。将类命名为 SportObject 类，并添加打篮球的成员方法 beatBasketball()。代码如下：

```
<?php
        class SportObject{
                function beatBasketball()($name,$height,$avoirdupois,$age,$sex){                //声明成员方法
                        echo "姓名：".$name;            //方法实现的功能
                        echo "身高：".$height;          //方法实现的功能
                        echo "年龄：".$age;             //方法实现的功能
                }
        }
?>
```

该方法的作用是输出申请打篮球人的基本信息，包括姓名、身高和年龄。这些信息是通过方法的参数传进来的。



14.2.3 类的实例化

类的方法已经添加，接下来就使用方法，但使用方法不像使用函数那么简单。首先要对类进行实例化，实例化是通过关键字 new 来声明一个对象。然后使用如下格式来调用要使用的方法：

```
“对象名 -> 成员方法”
```

在 14.1 节中已经讲过，类是一个抽象的描述，是功能相似的一组对象的集合。如果想用到类中的方法或变量，首先就要把它具体落实到一个实体，也就是对象上。

【例 14.1】 以 SportObject 类为例，实例化一个对象并调用方法 beatBasketball()。实例代码如下：**（实例位置：光盘\TM\sl\14\1）**

```
<?php
    class SportObject{
        function beatBasketball($name,$height,$avoirdupois,$age,$sex){     //声明成员方法
            if($height>180 and $avoirdupois<=100){
                return $name."，符合打篮球的要求!";                         //方法实现的功能
            }else{
                return $name."，不符合打篮球的要求!";                       //方法实现的功能
            }
        }
    }
    $sport=new SportObject();
    echo $sport->beatBasketball('明日','185','80','20 周岁','男');
?>
```

结果为：明日，符合打篮球的要求!



说明

实例 14.1 创建了图 14.1 中的运动类，同时也完成了图 14.2 中对类的实例化操作，最终输出方法判断的结果。

14.2.4 成员变量

类中的变量，也称为成员变量（也有称为属性或字段的）。成员变量用来保存信息数据，或与成员方法进行交互来实现某项功能。

定义成员变量的格式为：

```
关键字 成员变量名。
```



说明

关键字可以使用 public、private、protected、static 和 final 中的任意一个。在 14.2.9 节之前，所有的实例都使用 public 来修饰。对于关键字的使用，将在 14.2.9 节和 14.2.10 节中进行介绍。

访问成员变量和访问成员方法是一样的。只要把成员方法换成成员变量即可，格式为：

```
对象名 -> 成员变量
```

【例 14.2】 以图 14.1 和图 14.2 中描述的类和类的实例化为例，将其通过代码实现。首先定义运动类 SportObject，声明 5 个成员变量$name、$height、$avoirdupois、$age 和$sex。然后定义一个成员方法 bootFootball()，用于判断申请的运动员是否适合这个运动项目。最后，实例化类，通过实例化返回对象调用指定的方法，根据运动员填写的参数，判断申请的运动员是否符合要求。实例代码如下：**（实例位置：光盘\TM\sl\14\2）**

```
<?php
class SportObject{
	public $name;                                           //定义成员变量
	public $height;                                         //定义成员变量
	public $avoirdupois;                                    //定义成员变量

	public function bootFootBall($name,$height,$avoirdupois){  //声明成员方法
		$this->name=$name;
		$this->height=$height;
		$this->avoirdupois=$avoirdupois;
		if($this->height<185 and $this->avoirdupois<85){
			return $this->name."，符合踢足球的要求!";        //方法实现的功能
		}else{
			return $this->name."，不符合踢足球的要求!";      //方法实现的功能
		}
	}
}
$sport=new SportObject();                               //实例化类，并传递参数
echo $sport->bootFootBall('明日','185','80');            //执行类中的方法
?>
```

结果为：明日，不符合踢足球的要求!

说明

$this -> 作用是调用本类中的成员变量或成员方法，这里只要知道含义即可。在 14.2.8 节中将介绍相关的知识。

注意

无论是使用“$this->”还是使用“对象名->”的格式，后面的变量是没有$符号的，如$this- > beatBasketBall、$sport-> beatBasketBall。这是一个出错几率很高的错误。



14.2.5　类常量

既然有变量，当然也会有常量。常量就是不会改变的量，是一个恒值。圆周率是众所周知的一个常量。定义常量使用关键字 const，如：

```
const PI= 3.14159;
```

【例 14.3】 本例先声明一个常量，又声明了一个变量，实例化对象后分别输出两个值。实例代码如下：（**实例位置：光盘\TM\sl\14\3**）

```
<?php
class SportObject{
	const BOOK_TYPE = '计算机图书';				//声明常量 BOOK_TYPE
	public $object_name;					//声明变量，用来存放商品名称
	function setObjectName($name){				//声明方法 setObjectName()
		$this -> object_name = $name;			//设置成员变量值
	}
	function getObjectName(){				//声明方法 getObjectName()
		return $this -> object_name;
	}
}
$c_book = new SportObject();					//实例化对象
$c_book -> setObjectName("PHP 类");				//调用方法 setObjectName()
echo SportObject::BOOK_TYPE." -> ";				//输出常量 BOOK_TYPE
echo $c_book -> getObjectName();				//调用方法 getObjectName()
?>
```

结果为：计算机图书 -> PHP 类

可以发现，常量的输出和变量的输出是不一样的。常量不需要实例化对象，直接由“类名+常量名”调用即可。常量输出的格式为：

```
类名::常量名
```


说明

类名和常量名之间的两个冒号“::”称为作用域操作符，使用这个操作符可以在不创建对象的情况下调用类中的常量、变量和方法。关于作用域操作符，将在 14.2.8 节中进行介绍。

14.2.6　构造方法和析构方法

1．构造方法

当一个类实例化一个对象时，可能会随着对象初始化一些成员变量。如例 14.2 中的 SportObject 类，现在再添加一些成员变量，类的形式如下：

```
class SportObject{
        public $name;                                         //定义姓名成员变量
        public $height;                                       //定义身高成员变量
        public $avoirdupois;                                  //定义体重成员变量
        public $age;                                          //定义年龄成员变量
        public $sex;                                          //定义性别成员变量
}
?>
```

声明一个 SportObject 类的对象，并对这个类的一些成员变量赋初值。代码如下：

```
$sport=new SportObject('明日','185','80','20','男');          //实例化类，并传递参数
$sport ->name="明日 ";                                        //为成员变量赋值
$sport ->height=185;                                          //为成员变量赋值
$sport ->avoirdupois=80;                                      //为成员变量赋值
$sport ->age=20;                                              //为成员变量赋值
$sport ->sex="男";                                            //为成员变量赋值
echo $sport->bootFootBall();                                  //执行方法
```

可以看到，如果赋初值比较多，写起来就比较麻烦。为此，PHP 引入了构造方法。构造方法是生成对象时自动执行的成员方法，作用就是初始化对象。该方法可以没有参数，也可以有多个参数。构造方法的格式如下：

```
void __construct([mixed args [,…]])
```

注意

函数中的“__”是两条下划线“_”。

【例 14.4】 本例重写了 SportObject 类和 bootFootBall()对象，下面通过具体实例查看重写后的对象在使用上有哪些不一样。实例代码如下：**（实例位置：光盘\TM\sl\14\4）**

```
<?php
class SportObject{
        public $name;                                              //定义成员变量
        public $height;                                            //定义成员变量
        public $avoirdupois;                                       //定义成员变量
        public $age;                                               //定义成员变量
        public $sex;                                               //定义成员变量
        public function __construct($name,$height,$avoirdupois,$age,$sex){        //定义构造方法
                $this->name=$name;                                 //为成员变量赋值
                $this->height=$height;                             //为成员变量赋值
                $this->avoirdupois=$avoirdupois;                   //为成员变量赋值
                $this->age=$age;                                   //为成员变量赋值
                $this->sex=$sex;                                   //为成员变量赋值
        }
        public function bootFootBall(){                            //声明成员方法
                if($this->height<185 and $this->avoirdupois<85){
                        return $this->name."，符合踢足球的要求!";      //方法实现的功能
```

```
        }else{
            return $this->name."，不符合踢足球的要求!";          //方法实现的功能
        }
    }
}
$sport=new SportObject('明日','185','80','20','男');                //实例化类，并传递参数
echo $sport->bootFootBall();                                       //执行类中的方法
```

结果为：明日，不符合踢足球的要求!

可以看到，重写后的类，在实例化对象时只需一条语句即可完成赋值。

说明

构造方法是初始化对象时使用的。如果类中没有构造方法，那么 PHP 会自动生成一个。自动生成的构造方法没有任何参数，没有任何操作。

2. 析构方法

析构方法的作用和构造方法正好相反，是对象被销毁时被调用的，作用是释放内存。析构方法的格式为：

```
void __destruct ( void )
```

【例 14.5】 本实例首先声明一个对象 car，然后再销毁对象。可以看出，使用析构方法十分简单。实例代码如下：（**实例位置：光盘\TM\sl\14\5**）

```
<?php
class SportObject{
    public $name;                                               //定义姓名成员变量
    public $height;                                             //定义身高成员变量
    public $avoirdupois;                                        //定义体重成员变量
    public $age;                                                //定义年龄成员变量
    public $sex;                                                //定义性别成员变量
    public function __construct($name,$height,$avoirdupois,$age,$sex){     //定义构造方法
        $this->name=$name;                                      //为成员变量赋值
        $this->height=$height;                                  //为成员变量赋值
        $this->avoirdupois=$avoirdupois;                        //为成员变量赋值
        $this->age=$age;                                        //为成员变量赋值
        $this->sex=$sex;                                        //为成员变量赋值
    }
    public function bootFootBall(){                             //声明成员方法
        if($this->height<185 and $this->avoirdupois<85){
            return $this->name."，符合踢足球的要求!";            //方法实现的功能
        }else{
            return $this->name."，不符合踢足球的要求!";          //方法实现的功能
        }
    }
    function __destruct(){                                      //析构函数
        echo "<p><b>对象被销毁，调用析构函数。</b></p>";
```

```
    }
}
$sport=new SportObject('明日','185','80','20','男');                    //实例化类，并传递参数
//unset($sport);
?>
```

结果为：对象被销毁，调用析构函数。

说明

PHP 使用的是一种“垃圾回收”机制，自动清除不再使用的对象，释放内存。就是说即使不使用 unset 函数，析构方法也会自动被调用，这里只是明确一下析构函数在何时被调用。一般情况下是不需要手动创建析构方法的。

14.2.7 继承和多态的实现

继承和多态最根本的作用就是完成代码的重用。下面就来介绍 PHP 的继承和多态。

1. 继承

子类继承父类的所有成员变量和方法，包括构造函数，当子类被创建时，PHP 会先在子类中查找构造方法。如果子类有自己的构造方法，PHP 会先调用子类中的方法。当子类中没有时，PHP 则去调用父类中的构造方法，这就是继承。

例如，在 14.1 节中通过图片展示了一个运动类，在这个运动类中包含很多个方法，代表不同的体育项目，各种体育项目的方法中有公共的属性。例如，姓名、性别、年龄……但还会有许多不同之处，例如，篮球对身高的要求、举重对体重的要求……如果都由一个 SportObject 类来生成各个对象，除了那些公共属性外，其他属性和方法则需自己手动来写，工作效率得不到提高。这时，可以使用面向对象中的继承来解决这个难题。

下面来看如何通过 PHP 中的继承来解决上述问题。继承是通过关键字 extends 来声明的，继承的格式如下：

```
class subClass extends superClass{
    …
}
```

说明

subClass 为子类名称，superClass 为父类名称。

【例 14.6】 本例用 SportObject 类生成了两个子类：BeatBasketBall 和 WeightLifting，两个子类使用不同的构造方法实例化了两个对象 beatbasketball 和 weightlifting，并输出信息。实例代码如下：（**实例位置：光盘\TM\sl\14\6**）



```
<?php
/*   父类   */
class SportObject{
        public $name;                                           //定义姓名成员变量
        public $age;                                            //定义年龄成员变量
        public $avoirdupois;                                    //定义体重成员变量
        public $sex;                                            //定义性别成员变量
        public function __construct($name,$age,$avoirdupois,$sex){          //定义构造方法
                $this->name=$name;                              //为成员变量赋值
                $this->age=$age;                                //为成员变量赋值
                $this->avoirdupois=$avoirdupois;                //为成员变量赋值
                $this->sex=$sex;                                //为成员变量赋值
        }
        function showMe(){                                      //在父类中定义方法
                echo '这句话不会显示。';
        }
}
/*   子类 BeatBasketBall   */
class BeatBasketBall extends SportObject{                       //定义子类，继承父类
        public $height;                                         //定义身高成员变量
        function __construct($name,$height){                    //定义构造方法
                $this -> height = $height;                      //为成员变量赋值
                $this -> name = $name;                          //为成员变量赋值
        }
        function showMe(){                                      //定义方法
                if($this->height>185){
                        return $this->name."，符合打篮球的要求!";          //方法实现的功能
                }else{
                        return $this->name."，不符合打篮球的要求!";        //方法实现的功能
                }
        }
}
/*   子类 WeightLifting   */
class WeightLifting extends SportObject{                        //继承父类
        function showMe(){                                      //定义方法
                if($this->avoirdupois<85){
                        return $this->name."，符合举重的要求!";            //方法实现的功能
                }else{
                        return $this->name."，不符合举重的要求!";          //方法实现的功能
                }
        }
}
//实例化对象
$beatbasketball = new BeatBasketBall('科技','190');              //实例化子类
$weightlifting = new WeightLifting('明日','185','80','20','男');
echo $beatbasketball->showMe()."<br>";                          //输出结果
echo $weightlifting->showMe()."<br>";
?>
```

运行结果如图 14.3 所示。

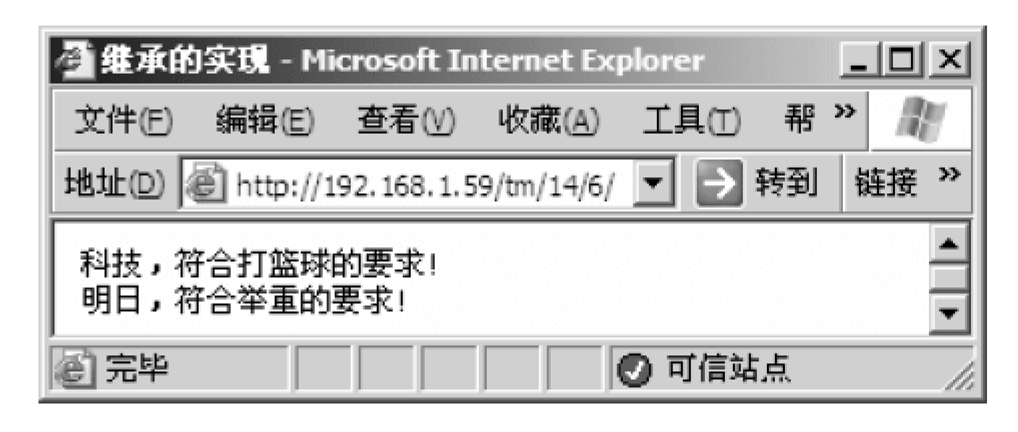


图 14.3　继承的实现

2．多态

多态好比有一个成员方法让大家去游泳，这个时候有的人带游泳圈，还有人拿浮板，还有人什么也不带。虽是同一种方法，却产生了不同的形态，就是多态。

多态存在两种形式：覆盖和重载。

（1）所谓覆盖就是在子类中重写父类的方法，而在两个子类的对象中虽然调用的是父类中相同的方法，但返回的结果是不同的。例如，在实例 14.6 中，在两个子类中都调用了父类中的方法 showMe()，但是返回的结果却不同。

（2）重载，是类的多态的另一种实现。函数重载指一个标识符被用作多个函数名，且能够通过函数的参数个数或参数类型将这些同名的函数区分开来，调用不发生混淆。其好处是可实现代码重用，不用为了对不同的参数类型或参数个数而写多个函数。

多个函数使用同一个名字，但参数个数、参数数据类型不同。调用时，虽然方法名称相同，但根据参数个数或者参数数据类型不同自动调用对应的函数。

下面看一个重载的简单实例，根据传递的参数个数不同，调用不同的方法，返回不同的值。

```
<?php
	class C{
		function __call($name,$num){							//调用不存在的方法
			echo "方法名称：" . $name . "<p>";				//输出方法名
			echo "参数存在个数：" . count($num) . "<p>";		//输出参数个数
			if (count($num) == 1){							//根据参数个数调用不同的方法
				echo $this->list1($a);
			}
			if (count($num) == 2){							//根据参数个数调用不同的方法
				echo $this->list2($a,$b);
			}
		}
		public function list1($a){							//定义方法
			return "这是 list1 函数";
		}
		public function list2($a,$b){						//定义方法
			return "这是 list2 函数";
		}
	}
	$a = new C;												//类的实例化
	$a->listshow(1,2);										//调用方法，传递参数
?>
```



14.2.8　"$this ->"和"::"的使用

通过例 14.6 可以发现，子类不仅可以调用自己的变量和方法，也可以调用父类中的变量和方法。那么对于其他不相关的类成员同样可以调用。

PHP 是通过伪变量"$this ->"和作用域操作符"::"来实现这些功能的，这两个符号在前面的学习中都有过简单的介绍。本节将详细讲解两者的使用。

1．$this->

在 14.2.3 节"类的实例化"中，对如何调用成员方法有了基本的了解，那就是用对象名加方法名，格式为"对象名->方法名"。但在定义类时（如 SportObject 类），根本无法得知对象的名称是什么。这时如果想调用类中的方法，就要用伪变量$this->。$this 的意思就是本身，所以$this->只可以在类的内部使用。

【例 14.7】 当类被实例化后，$this 同时被实例化为本类的对象，这时对$this 使用 get_class()函数，将返回本类的类名。实例代码如下：**（实例位置：光盘\TM\sl\14\7）**

```
<?php
	class example{										//创建类 example
		function exam(){								//创建成员方法
			if(isset($this)){							//判断变量$this 是否存在
				echo '$this 的值为：'.get_class($this);		//如果存在，输出$this 所属类的名字
			}else{
				echo '$this 未定义';
			}
		}
	}
	$class_name = new example();						//实例化对象$class_name
	$class_name->exam();								//调用方法 exam()
?>
```

结果为，$this 的值为：example



get_class()函数返回对象所属类的名字，如果不是对象，则返回 false。

2．操作符"::"

相比伪变量$this 只能在类的内部使用，操作符"::"更为强大。操作符"::"可以在没有声明任何实例的情况下访问类中的成员方法或成员变量。使用"::"操作符的通用格式为：

```
关键字::变量名/常量名/方法名
```

这里的关键字分为以下 3 种情况。

☑　parent 关键字：可以调用父类中的成员变量、成员方法和常量。

☑ self 关键字：可以调用当前类中的静态成员和常量。

☑ 类名：可以调用本类中的变量、常量和方法。

【例 14.8】 本例依次使用了类名、parent 关键字和 self 关键字来调用变量和方法。读者可以观察输出的结果。实例代码如下：**（实例位置：光盘\TM\sl\14\8）**

```
<?php
class Book{
	const NAME = 'computer';	//常量 NAME
	function __construct(){	//构造方法
		echo '本月图书类冠军为：'.Book::NAME.' ';	//输出默认值
	}
}
class l_book extends Book{	//Book 类的子类
	const NAME = 'foreign language';	//声明常量
	function __construct(){	//子类的构造方法
		parent::__construct();	//调用父类的构造方法
		echo '本月图书类冠军为：'.self::NAME.' ';	//输出本类中的默认值
	}
}
$obj = new l_book();	//实例化对象
?>
```

结果为，本月图书类冠军为：computer 本月图书类冠军为：foreign language


说明

关于静态方法（变量）的声明及使用可参考 14.2.10 节相关内容。

14.2.9 数据隐藏

细心的读者看到这里，一定会有一个疑问：面向对象编程的特点之一是封装性，即数据隐藏。可在前面的学习中并没有突出这一点。对象中的所有变量和方法可以随意调用，甚至不用实例化也可以使用类中的方法、变量。这就是面向对象吗？

这当然不算是真正的面向对象。如果读者是从本章第一节来开始学习的，一定还会记得在 14.2.4 节讲成员变量时所提到的那几个关键字：public、private、protected、static 和 final。这就是用来限定类成员（包括变量和方法）的访问权限的。本节先来学习前 3 个。

说明

成员变量和成员方法在关键字的使用上都是一样的。这里只以成员变量为例说明几种关键字的不同用法。对于成员方法同样适用。

1. public（公共成员）

顾名思义，就是可以公开的、没有必要隐藏的数据信息。可以在程序中的任何位置（类内、类外）



被其他的类和对象调用。子类可以继承和使用父类中所有的公共成员。

在本章的前半部分，所有的变量都被声明为 public，而所有的方法在默认状态下也是 public。所以对变量和方法的调用显得十分混乱。为了解决这个问题，就需要使用第二个关键字 private。

2. private（私有成员）

被 private 关键字修饰的变量和方法，只能在所属类的内部被调用和修改，不可以在类外被访问。在子类中也不可以。

【例 14.9】 在本例中，对私有变量$name 的修改与访问，只能通过调用成员方法来实现。如果直接调用私有变量，将会发生错误。实例代码如下：**（实例位置：光盘\TM\sl\14\9）**

```
<?php
class Book{
	private $name = 'computer';                 //声明私有变量$name
	public function setName($name){             //设置私有变量方法
		$this -> name = $name;
	}
	public function getName(){                  //读取私有变量方法
		return $this -> name;
	}
}
class LBook extends Book{                       //Book 类的子类
}
$lbook = new LBook();                           //实例化对象
echo '正确操作私有变量的方法：';                  //正确操作私有变量
$lbook -> setName("PHP5 从入门到应用开发");
echo $lbook -> getName();
echo '<br>直接操作私有变量的结果：';              //错误操作私有变量
echo Book::$name;
?>
```

运行结果如图 14.4 所示。



图 14.4　private 关键字



说明

对于成员方法，如果没有写关键字，那么默认就是 public。从本节开始，以后所有的方法及变量都会带上关键字，这是一种良好的书写习惯。

3．protected（保护成员）

private 关键字可以将数据完全隐藏起来，除了在本类外，其他地方都不可以调用，子类也不可以。对于有些变量希望子类能够调用，但对另外的类来说，还要做到封装。这时，就可以使用 protected。

被 protected 修饰的类成员，可以在本类和子类中被调用，其他地方则不可以被调用。

【例 14.10】 本实例首先声明一个 protected 变量，然后使用子类中的方法调用一次，最后在类外直接调用一次，观察一下运行结果。实例代码如下：**（实例位置：光盘\TM\sl\14\10）**

```
<?php
class Book{
	protected $name = 'computer';			//声明保护变量$name
}
class LBook extends Book{				//Book 类的子类
	public function showMe(){
		echo '对于 protected 修饰的变量，在子类中是可以直接调用的。如：$name = '.$this -> name;
	}
}
$lbook = new LBook();					//实例化对象
$lbook -> showMe();
echo '<p>但在其他的地方是不可以调用的，否则：';		//对私有变量进行操作
$lbook -> name = 'history';
?>
```

运行结果如图 14.5 所示。

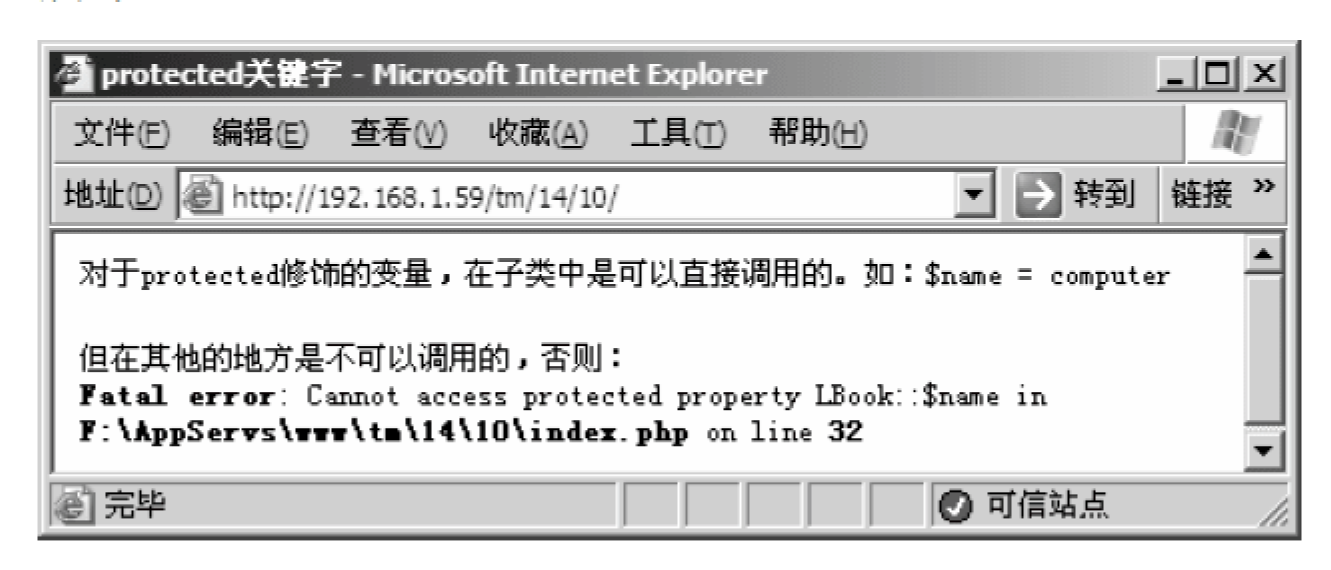


图 14.5　protected 关键字



说明

虽然 PHP 中没有对修饰变量的关键字做强制性的规定和要求，但从面向对象的特征和设计方面考虑，一般使用 private 或 protected 关键字来修饰变量，以防止变量在类外被直接修改和调用。

14.2.10　静态变量（方法）

不是所有的变量（方法）都要通过创建对象来调用。可以通过给变量（方法）加上 static 关键字来直接调用。调用静态成员的格式为：



```
关键字::静态成员
```

关键字可以是：

☑　self，在类内部调用静态成员时所使用。

☑　静态成员所在的类名，在类外调用类内部的静态成员时所用。



注意

在静态方法中，只能调用静态变量，而不能调用普通变量，而普通方法则可以调用静态变量。

使用静态成员，除了可以不需要实例化对象，另一个作用就是在对象被销毁后，仍然保存被修改的静态数据，以便下次继续使用。这个概念比较抽象，下面结合一个实例说明。

【例 14.11】 本例首先声明一个静态变量$num，声明一个方法，在方法的内部调用静态变量，然后给变量加 1。依次实例化这个类的两个对象，并输出方法。可以发现两个对象中的方法返回的结果有了一些联系。直接使用类名输出静态变量，看有什么效果。实例代码如下：（**实例位置：光盘\TM\sl\14\11**）

```
<?php
class Book{                                    //Book 类
    static $num = 0;                           //声明一个静态变量$num，初值为 0
    public function showMe(){                  //声明一个方法
        echo '您是第'.self::$num.'位访客';       //输出静态变量
        self::$num++;                          //将静态变量加 1
    }
}
$book1 = new Book();                           //实例化对象$book1
$book1 -> showMe();                            //调用对象$book1 的 showMe()方法
echo "<br>";
$book2 = new Book();                           //实例化对象$book2;
$book2 -> showMe();                            //调用对象$book2 的 showMe()方法
echo "<br>";
echo '您是第'.Book::$num.'为访客';               //直接使用类名调用静态变量
?>
```

运行结果如图 14.6 所示。



图 14.6　静态变量的使用

如果将程序代码中的静态变量改为普通变量，如“private $num = 0;”，那么结果就不一样了。读者可以动手试一试。


说明

静态成员不用实例化对象，当类第一次被加载时就已经分配了内存空间，所以直接调用静态成员的速度要快一些。但如果静态成员声明得过多，空间一直被占用，反而会影响系统的功能。这个尺度只能通过实践积累，才能真正地掌握。

14.3　PHP 对象的高级应用

视频讲解：光盘\TM\lx\14\面向对象的高级应用.exe

经过 14.2 节的学习，相信读者对 PHP 的面向对象已经有了一定的了解。下面来学习一些面向对象的高级应用。

14.3.1　final 关键字

final，中文含义是“最终的”、“最后的”。被 final 修饰过的类和方法就是“最终的版本”。

如果有一个类的格式为：

```
final class class_name{
//…
}
```

说明该类不可以再被继承，也不能再有子类。

如果有一个方法的格式为：

```
final function method_name()
```

说明该方法在子类中不可以进行重写，也不可以被覆盖。

【例 14.12】 本例为 SportObject 类设置关键字 final，并生成一个子类 MyBook。可以看到程序报错，无法执行。实例代码如下：（**实例位置：光盘\TM\sl\14\12**）

```
<?php
	final class SportObject{                                        //final 类 SportObject
		function	__construct(){                              //构造方法
			echo 'initialize object';
		}
	}
	class MyBook extends SportObject{                           //创建 SportObject 的子类 Mybook
		static function exam(){                                 //子类中的方法
			echo "You can't see me";
		}
	}
```



```
        MyBook::exam();                                           //调用子类方法
?>
```

结果为：

```
Fatal error: Class MyBook may not inherit from final class (SportObject) in F:\AppServ\www\tm\14\12\index.php
on line 30
```

14.3.2　抽象类

抽象类是一种不能被实例化的类，只能作为其他类的父类来使用。抽象类使用 abstract 关键字来声明，格式为：

```
abstract class AbstractName{
…
}
```

抽象类和普通类相似，包含成员变量、成员方法。两者的区别在于，抽象类至少要包含一个抽象方法。抽象方法没有方法体，其功能的实现只能在子类中完成。抽象方法也是使用 abstract 关键字来修饰的，它的格式为：

```
abstract function abstractName();
```


注意

在抽象方法后面要有分号“;”。

抽象类和抽象方法主要应用于复杂的层次关系中，这种层次关系要求每一个子类都包含并重写某些特定的方法。举一个例子，中国的美食是多种多样的，有吉菜、鲁菜、川菜、粤菜等。每种菜系使用的都是煎、炒、烹、炸等手法，只是在具体的步骤上，各有各的不同。如果把中国美食当作一个大类 Cate，下面的各大菜系就是 Cate 的子类，而煎、炒、烹、炸则是每个类中都有的方法。每个方法在子类中的实现都是不同的，在父类中无法规定。为了统一规范，不同子类的方法要有一个相同的方法名：decoct（煎）、stir_fry（炒）、cook（烹）、fry（炸）。

【例 14.13】 下面实现一个商品抽象类 CommodityObject，该抽象类包含一个抽象方法 service()。为抽象类生成两个子类 MyBook 和 MyComputer，分别在两个子类中实现抽象方法。最后实例化两个对象，调用实现后的抽象方法，输出结果。实例代码如下：**（实例位置：光盘\TM\sl\14\13）**

```
<?php
        abstract class CommodityObject{                                    //定义抽象类
                abstract function service($getName,$price,$num);           //定义抽象方法
        }
        class MyBook extends CommodityObject{                              //定义子类，继承抽象类
                function service($getName,$price,$num){                    //定义方法
                        echo '您购买的商品是'.$getName.'，该商品的价格是：'.$price.' 元。';
                        echo '您购买的数量为：'.$num.' 本。';
                        echo '如发现缺页，损坏请在 3 日内更换。';
```

```
        }
    }
    class MyComputer extends CommodityObject{                    //定义子类继承父类
        function service($getName,$price,$num){                  //定义方法
            echo '您购买的商品是'.$getName.'，该商品的价格是：'.$price.' 元。';
            echo '您购买的数量为：'.$num.' 台。';
            echo '如发生非人为质量问题，请在 3 个月内更换。';
        }
    }
    $book = new MyBook();                                        //实例化子类
    $computer = new MyComputer();                                //实例化子类
    $book -> service('《PHP 从入门到精通》',85,3);                 //调用方法
    echo '<p>';
    $computer -> service('XX 笔记本',8500,1);                     //调用方法
?>
```

运行结果如图 14.7 所示。



图 14.7　抽象类

14.3.3　接口的使用

继承特性简化了对象、类的创建，增加了代码的可重性。但 PHP 只支持单继承。如果想实现多重继承，就要使用接口。PHP 可以实现多个接口。

接口类通过 interface 关键字来声明，并且类中只能包含未实现的方法和一些成员变量，格式如下：

```
interface InterfaceName{
    function interfaceName1();
    function interfaceName2();
    …
}
```

注意

不要用 public 以外的关键字来修饰接口中的类成员，对于方法，不写关键字也可以。这是一个接口类自身的属性决定的。

子类是通过 implements 关键字来实现接口的，如果要实现多个接口，那么每个接口之间应使用逗号“,”连接。而且所有未实现的方法需要在子类中全部实现，否则 PHP 将会出现错误。格式如下：



```
class SubClass implements InterfaceName1，InterfaceName2{
        function interfaceName1(){
                //功能实现
        }
        function interfaceName2(){
                //功能实现
        }
        …
}
```

【例 14.14】 本例首先声明了两个接口 MPopedom 和 MPurview，接着声明了两个类 Member 和 Manager，其中 Member 类继承了 MPopedom 接口；Manager 继承了 MPopedom 和 MPurview 接口。分别实现各自的成员方法后，实例化两个对象$member 和$manager。最后调用实现后的方法。实例代码如下：（**实例位置：光盘\TM\sl\14\14**）

```
<?php
        /*  声明接口 MPopedom  */
        interface MPopedom{
                function popedom();
        }
        /*  声明接口 MPurview  */
        interface MPurview{
                function purview();
        }
        /*  创建子类 Member，实现一个接口 MPurview  */
        class Member implements MPurview{
                function purview(){
                        echo '会员拥有的权限。';
                }
        }
        /*  创建子类 Manager，实现多个接口 MPurview 和 MPopedom  */
        class Manager implements MPurview,MPopedom{
                function purview(){
                        echo '管理员拥有会员的全部权限。';
                }
                function popedom(){
                        echo '管理员还有会员没有的权限';
                }
        }
        $member = new Member();                         //类 Member 实例化
        $manager = new Manager();                       //类 Manager 实例化
        $member -> purview();                           //调用$member 对象的 purview 方法
        echo '<p>';
        $manager -> purview();                          //调用$manager 对象的 purview 方法
        $manager ->popedom();                           //调用$manager 对象的 popedom 方法
?>
```

运行结果如图 14.8 所示。



图 14.8　应用接口

通过上面的实例可以发现，抽象类和接口实现的功能十分相似。抽象类的优点是可以在抽象类中实现公共的方法，而接口则可以实现多继承。至于何时使用抽象类和接口就要看具体实现了。

14.3.4　克隆对象

1．克隆对象

在 PHP 4 中，对象被当作普通的数据类型来使用。如果想引用对象，需要使用“&”来声明，否则会按照 PHP 4 的默认方式来按值传递对象。下面结合实例说明。

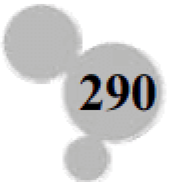
【例 14.15】 本例首先实例化一个 SportObject 类的对象$book1，$book1 的默认值为 book，然后将对象$book1 使用普通数据类型的赋值方式给对象$book2 赋值。改变$book2 的值为 computer，再输出对象$book1 的值。实例代码如下：**（实例位置：光盘\TM\sl\14\15）**

```
<?php
class SportObject{                                   //类 SportObject
	private $object_type = 'book';                   //声明私有变量$object_type，并赋初值为 book
	public function setType($type){                  //声明成员方法 setType，为变量$object_type 赋值
		$this -> object_type = $type;
	}
	public function getType(){                       //声明成员方法 getType，返回变量$object_type 的值
		return $this -> object_type;
	}
}
$book1 = new SportObject();                          //实例化对象$book1
$book2 = $book1;                                     //使用普通数据类型的方法给对象$book2 赋值
$book2 -> setType('computer');                       //改变对象$book2 的值
echo '对象$book1 的值为：'.$book1 -> getType();      //输出对象$book1 的值
?>
```

上面的实例在 PHP 5 中的返回值为“对象$book1 的值为：computer”，因为$book2 只是$book1 的一个引用；而在 PHP 4 中的返回值是“对象$book1 的值为：book”，因为对象$book2 是$book1 的一个备份。

在 PHP 5 中如果需要将对象复制，也就是克隆一个对象。需要使用关键字 clone 来实现。克隆对象的格式为：

```
$object1 = new ClassName();
$object2 = clone $object1;
```



将例 14.15 中的$book2=$book1 修改为$book2=clone $book1，其他不变，即可返回 PHP 4 中的结果。

2. __clone()方法

有时除了单纯地克隆对象外，还需要克隆出来的对象可以拥有自己的属性和行为。这时就可以使用__clone()方法来实现。__clone()方法的作用是：在克隆对象的过程中，调用__clone()方法，可以使克隆出来的对象保持自己的一些行为及属性。

【例 14.16】 本实例将例 14.15 的代码做一些修改。在对象$book1 中创建__clone()方法，该方法实现的功能是将变量$object_type 的默认值从 book 修改为 computer。使用对象$book1 克隆出对象$book2，输出$book1 和$book2 的$object_type 值，查看最终的结果。实例代码如下：（**实例位置：光盘\TM\sl\14\16**）

```
<?php
class SportObject{                                      //类 SportObject
    private $object_type = 'book';                      //声明私有变量$object_type，并赋初值为 book
    public function setType($type){                     //声明成员方法 setType，为变量$object_type 赋值
        $this -> object_type = $type;
    }
    public function getType(){                          //声明成员方法 getType，返回变量$object_type 的值
        return $this -> object_type;
    }
    public function __clone(){                          //声明__clone()方法
        $this ->object_type = 'computer';               //将变量$object_type 的值修改为 computer
    }
}
$book1 = new SportObject();                             //实例化对象$book1
$book2 = clone $book1;                                  //使用普通数据类型的方法给对象$book2 赋值
echo '对象$book1 的变量值为：'.$book1 -> getType();     //输出对象$book1 的值
echo '<br>';
echo '对象$book2 的变量值为：'.$book2 -> getType();
?>
```

运行结果如图 14.9 所示。



图 14.9 __clone()方法

不难看出，对象$book2 克隆了对象$book1 的全部行为及属性，也拥有了属于自己的成员变量值。

14.3.5 对象比较

通过克隆对象，相信读者已经理解表达式$Object2 = $Object1 和$Object2 = clone $Object1 所表示的

不同含义。但在实际开发中，还需判断两个对象之间的关系是克隆还是引用，这时可以使用比较运算符“==”和“===”。两个等号“==”是比较两个对象的内容，3 个等号“===”是比较对象的引用地址。

【例 14.17】 本例首先实例化一个对象$book，然后分别创建一个克隆对象和引用，使用“==”和“===”判断它们之间的关系，最后输出结果。实例代码如下：（**实例位置：光盘\TM\sl\14\17**）

```
<?php
/*   SportObject 类   */
	class SportObject{
		private $name;
		function __construct($name){
			$this -> name = $name;
		}
	}
/***********************/
	$book = new SportObject('book');			//实例化一个对象$book
	$cloneBook = clone $book;				//克隆对象$cloneBook
	$referBook = $book;					//引用对象$referBook
	if($cloneBook == $book){				//使用==比较克隆对象和原对象
		echo '两个对象的内容相等<br>';
	}
	if($referBook === $book){				//使用===比较引用对象和原对象
		echo '两个对象的引用地址相等<br>';
	}
?>
```

结果为：两个对象的内容相等　两个对象的引用地址相等

14.3.6　对象类型检测

instanceof 操作符可以检测当前对象是属于哪个类。一般格式为：

```
ObjectName instanceof ClassName
```

【例 14.18】 本实例首先创建两个类，一个基类（SportObject）与一个子类（MyBook）。实例化一个子类对象，判断对象是否属于该子类，再判断对象是否属于基类。实例代码如下：（**实例位置：光盘\TM\sl\14\18**）

```
<?php
class SportObject{}						//创建空类 SportObject
class MyBook extends SportObject{				//创建子类 MyBook
	private $type;
}
$cBook = new MyBook();					//实例化对象$cBook
if($cBook instanceof MyBook)				//判断对象是否属于类 MyBook
	echo '对象$cBook 属于 MyBook 类<br>';
if($cBook instanceof SportObject)				//判断对象是否属于类 SportObject
```

```
	echo '对象$Book 属于 SportObject 类<br>';
?>
```

结果为：

```
对象$cBook 属于 MyBook 类
对象$cBook 属于 SportObject 类
```

14.3.7　魔术方法(__)

PHP 中有很多以两个下划线开头的方法，如前面已经介绍过的__construct()、__destruct()和__clone()，这些方法被称为魔术方法。本节中将会学习到其他一些魔术方法。

注意

PHP 中保留了所有以“__”开头的方法，所以只能使用在 PHP 文档中已有的这些方法，不要自己创建。

1. __set()和__get()方法

这两个魔术方法的作用分别为：

- ☑ 当程序试图写入一个不存在或不可见的成员变量时，PHP 就会执行__set()方法。__set()方法包含两个参数，分别表示变量名称和变量值，两个参数不可省略。
- ☑ 当程序调用一个未定义或不可见的成员变量时，可以通过__get()方法来读取变量值。__get()方法有一个参数，表示要调用的变量名。

注意

如果希望 PHP 调用这些魔术方法，首先必须在类中进行定义，否则 PHP 不会执行未创建的魔术方法。

【例 14.19】 本例首先声明类 SportObject，在类中创建一个私有变量$type 和两个魔术方法__set()、__get()，接着实例化一个对象$MyComputer，先对已存在的私有变量进行赋值和调用，再对未声明的变量$name 进行调用，最终查看输出结果。实例代码如下：**（实例位置：光盘\TM\sl\14\19）**

```
<?php
class SportObject{                                                    //类 SportObject
	private $type = ' ';                                              //私有变量$type
	private function __get($name){                                    //声明魔术方法__get()
		if(isset($this ->$name)){                                     //判断变量是否被声明
			echo '变量'.$name.'的值为：'.$this -> $name.'<br>';
		}else{
			echo '变量'.$name.'未定义，初始化为 0<br>';
			$this -> $name = 0;                                       //如果未被声明，则对变量初始化
		}
```

```
    }
    private function __set($name, $value){                  //声明魔术方法__set()
        if(isset($this -> $name)){                          //判断变量是否定义
            $this -> $name = $value;
            echo '变量'.$name.'赋值为：'.$value.'<br>';
        }else{
            $this -> $name = $value;                        //如果未定义，继续对变量进行赋值
            echo '变量'.$name.'被初始化为：'.$value.'<br>';  //输出警告信息
        }
    }
}
$MyComputer = new SportObject();                            //实例化对象$MyComputer
$MyComputer -> type = 'DIY';                                //给变量赋值
$MyComputer -> type;                                        //调用变量$type
$MyComputer -> name;                                        //调用变量$name
?>
```

运行结果如图 14.10 所示。

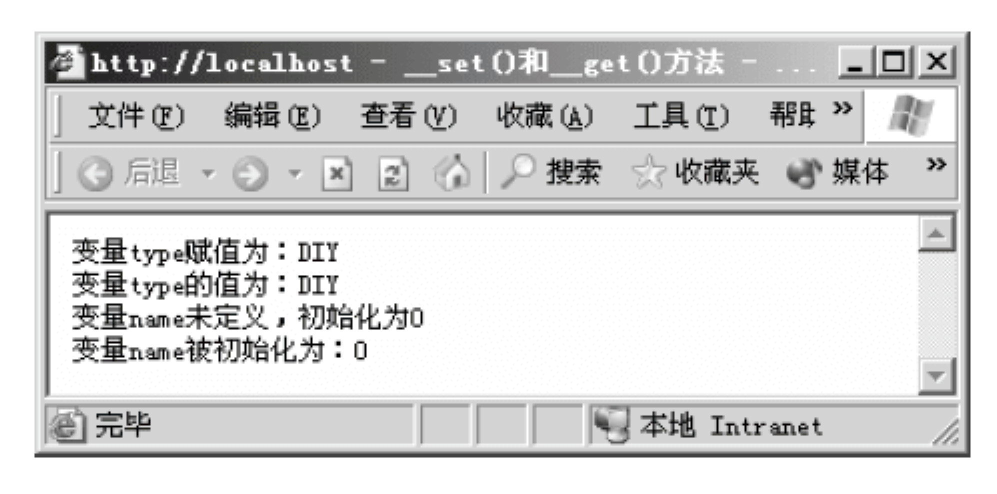


图 14.10　__set()和__get()方法

2．__call()方法

__call()方法的作用是：当程序试图调用不存在或不可见的成员方法时，PHP 会先调用__call()方法来存储方法名及其参数。__call()方法包含两个参数，即方法名和方法参数。其中，方法参数是以数组形式存在的。

【例 14.20】 本例声明一个类 SportObject，类中包含两个方法，即 myDream()和__call()。实例化对象$MyLife 需调用两个方法，一个是类中存在的 myDream()方法，一个是不存在的 mDream()方法。实例代码如下：（**实例位置：光盘\TM\sl\14\20**）

```
<?php
/*  类 SportObject  */
class SportObject{
    public function myDream(){                              //方法 myDream()
        echo '调用的方法存在，直接执行此方法。<p>';
    }
    public function __call($method, $parameter) {           //__call()方法
        echo '如果方法不存在，则执行__call()方法。<br>';
        echo '方法名为：'.$method.'<br>';                    //输出第一个参数，即方法名
        echo '参数有：';
        var_dump($parameter);                               //输出第二个参数，是一个参数数组
    }
```

```
}
$exam = new SportObject();                          //实例化对象$exam
$exam->myDream();                                   //调用存在的方法 myDream()
$exam -> mDream('how','what','why');                //调用不存在的方法 mDream()
?>
```

运行结果如图 14.11 所示。



图 14.11　__call()方法

3. __sleep()和__wakeup()方法

使用 serialize()函数可以实现序列化对象。就是将对象中的变量全部保存下来，对象中的类则只保存类名。在使用 serialize()函数时，如果实例化的对象包含__sleep()方法，则会先执行__sleep()方法。该方法可以清除对象并返回一个该对象中所有变量的数组。使用__sleep()方法的目的是关闭对象可能具有的数据库连接等类似的善后工作。

unserialize()函数可以重新还原一个被 serialize()函数序列化的对象，__wakeup()方法则是恢复在序列化中可能丢失的数据库连接及相关工作。

【例 14.21】 实例首先声明一个类 SportObject，类中有两个方法，即__sleep()和__wakeup()。实例化对象$myBook，使用 serialize()函数将对象序列化为一个字串$i，最后再使用 unserialize()函数将字串$i 还原为一个新对象。实例代码如下：（**实例位置：光盘\TM\sl\14\21**）

```
<?php
/*  创建对象 SportObject  */
class SportObject{
    private $type = 'DIY';                          //声明私有变量$type，初值为 DIY
    public function getType(){                      //声明 getType()方法，用来调用私有变量$type
        return $this -> type;                       //返回变量值
    }
    public function __sleep(){                      //声明魔术方法__sleep()
        echo '使用 serialize()函数将对象保存起来，可以存放到文本文件、数据库等地方<br>';
        return $this;
    }
    public function __wakeup(){                     //声明魔术方法__wakeup()
        echo '当需要该数据时，使用 unserialize()函数对已序列化的字符串进行操作，将其转换回对象<br>';
    }
}
$myBook = new SportObject();                        //实例化对象$myBook
$i = serialize($myBook);                            //序列化对象
echo '序列化后的字符串：'.$i.'<br>';                    //输出字串$i
```

```
$reBook = unserialize($i);                          //将字串$i 重新转换为对象$reBook
echo '还原后的成员变量：'.$reBook -> getType();     //调用新对象$reBook 的 getType()方法
?>
```

运行结果如图 14.12 所示。

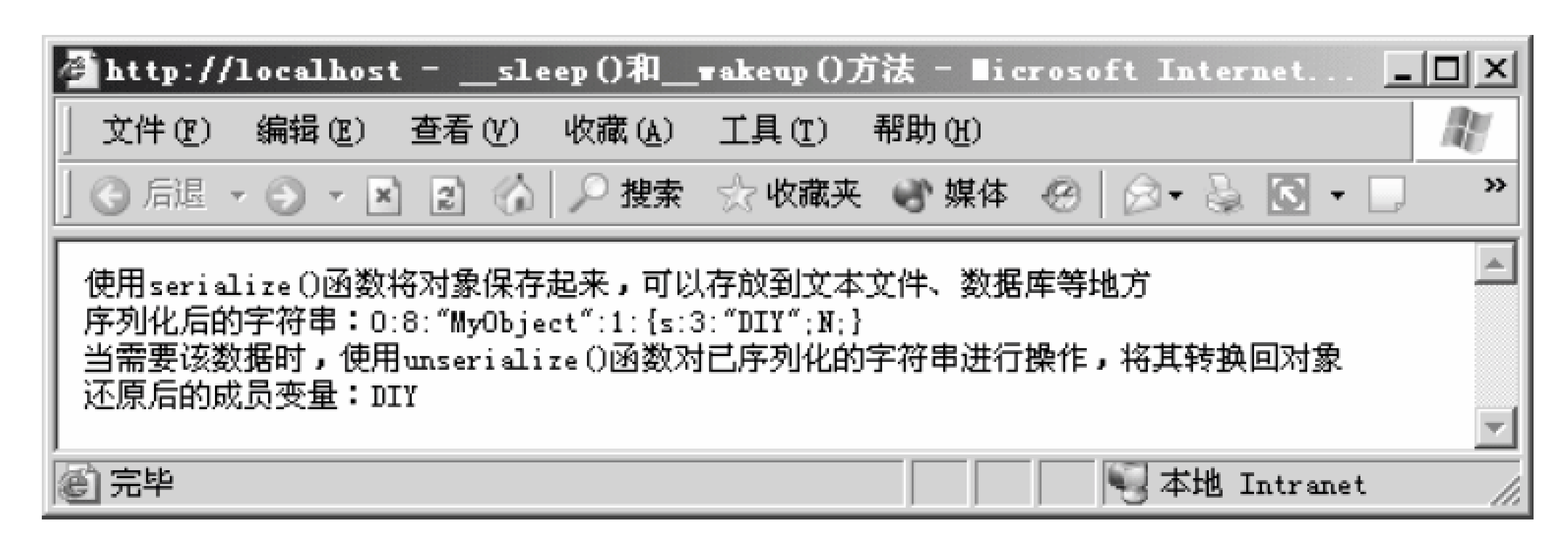


图 14.12　__sleep()和__wakeup()方法

4．__toString()方法

魔术方法__toString()的作用是：当使用 echo 或 print 输出对象时，将对象转化为字符串。

【例 14.22】 本例输出类 SportObject 的对象$myComputer，输出的内容为__toString()方法返回的内容。实例代码如下：**（实例位置：光盘\TM\sl\14\22）**

```
<?php
class SportObject{                                  //类 SportObject
	private $type = 'DIY';                          //声明私有变量$type
	public function __toString(){                   //声明__toString()方法
		return $this -> type;                       //方法返回私有变量$type 的值
	}
}
$myComputer = new SportObject();                    //实例化对象$myComputer
echo '对象$myComputer 的值为：';
echo $myComputer;                                   //输出对象$myComputer
?>
```

运行结果为，对象$myComputer 的值为：DIY

注意

（1）如果没有__toString()方法，直接输出对象将会发生致命错误（fatal error）。

（2）输出对象时应注意，echo 或 print 函数后面直接跟要输出的对象，中间不要加多余的字符，否则__toString 方法不会被执行。如 echo '字串'.$myComputer、echo ' '.$myComputer 等都不可以，一定要注意。

5．__autoload()方法

将一个独立、完整的类保存到一个 PHP 页中，并且文件名和类名保持一致，这是每个开发人员都需要养成的良好习惯。这样，在下次重复使用某个类时即能很轻易地找到它。但还有一个问题是让开发人员头疼不已的，如果要在一个页面中引进很多的类，需要使用 include_once()函数或 require_once()



函数一个个地引入。

PHP 5 解决了这个问题，__autoload()方法可以自动实例化需要使用的类。当程序要用到一个类，但该类还没有被实例化时，PHP 5 将使用__autoload()方法，在指定的路径下自动查找和该类名称相同的文件。如果找到，程序则继续执行；否则，报告错误。

【例 14.23】 本例首先创建一个类文件 SportObject.class.php，该文件包含类 SportObject。再创建 index 文件，在文件中先创建__autolaod()方法，手动实现查找的功能，如果查找成功，则使用 require_once()函数将文件动态引入。Index 文件的最后实例化对象参见输出结果。**（实例位置：光盘\TM\sl\14\23）**

类文件 SportObject.class.php 的代码如下：

```
<?php
class SportObject{                                              //声明类 SportObject
	private $cont;                                              //声明私有变量$cont
	public function __construct($cont){                         //创建构造方法
		$this -> cont = $cont;
	}
	public function __toString(){                               //创建__toString()方法
		return $this -> cont;
	}
}
?>
```

index.php 文件的代码如下：

```
<?php
	function __autoload($class_name){                           //创建__autoload()方法
		$class_path = $class_name.'.class.php';                 //类文件路径
		if(file_exists($class_path)){                           //判断类文件是否存在
			include_once($class_path);                          //动态包含类文件
		}else
			echo '类路径错误。';
	}
	$myBook = new SportObject("江山代有人才出　各领风骚数百年");    //实例化对象
	echo $myBook;                                               //输出类内容
?>
```

结果为：江山代有人才出　各领风骚数百年

14.4　面向对象的应用——中文字符串的截取类

本节将实现理论与实践的结合，将面向对象技术应用到实际的程序开发中。

为了确保程序页面整洁美观，经常需要对输出的字符串进行截取。在截取英文字符串时，可以使用 substr()函数来完成。但是当遇到中文字符串时，如果仍使用 substr()函数，那么就有可能出现乱码的情况，因为一个汉字是由两个字节组成的，所以当截取的内容出现单数时，就有可能将一个汉字给拆

分，从而导致输出一个不完整的汉字，也就是乱码。

【例 14.24】 本例编写 MsubStr 类，定义 csubstr()方法，实现对中文字符串的截取，避免在截取中文字符串时出现乱码的问题。实例代码如下：（**实例位置：光盘\TM\sl\14\24**）

```
<?php
class MsubStr{
    function csubstr($str, $start, $len) {  //$str 指定字符串，$start 指的是字符串的起始位置，$len 指的是长度
    $strlen = $start + $len;                //$strlen 字符串的总长度（从字符串的起始位置到字符串的总长度）
    for($i = 0; $i < $strlen; $i ++) {      //通过 for 循环语句，循环读取字符串
        if (ord ( substr ( $str, $i, 1 ) ) > 0xa0) {  //如果字符串中首个字节的 ASCII 序数值大于 0xa0，则表示为汉字
            $tmpstr .= substr ( $str, $i, 2 );  //每次取出两位字符赋给变量$tmpstr，即等于一个汉字
            $i ++;                              //变量自加 1
        } else {                                //如果不是汉字，则每次取出一位字符赋给变量$tmpstr
            $tmpstr .= substr ( $str, $i, 1 );
        }
    }
    return $tmpstr;                             //输出字符串
    }
}
$mc=new MsubStr();                              //类的实例化
?>
<table width="204" height="195" border="0" cellpadding="0" cellspacing="0" background="images/bg.JPG">
  <tr>
    <td><?php
            $string="关注明日科技，关注 PHP 从入门到精通改版！";
            if(strlen($string)>10){                         //判断字符串的长度
                echo substr($string,0,9)."...";             //截取字符串中 9 个字符
            }else{
                echo $string;
            }
        ?>
     </td>
  </tr>
  <tr>
    <td>
        <?php
            $strs="关注明日科技，关注 PHP 从入门到精通改版！";   //定义字符串
            if(strlen($string)>10){                         //判断字符串长度
                echo $mc ->csubstr($strs, "0" , "9")."...";  //应用类中的方法截取字符串
            }else{
                echo $strs;                                 //输出字符串
            }
        ?>
     </td>
  </tr>
  <tr>
    <td><?php
            $strs="关注 PHP 编程词典！";
            if(strlen($string)>30){
```

```
                    echo $mc ->csubstr($strs, "0" , "20")."...";
                }else{
                    echo $strs;
                }
            ?>
        </td>
    </tr>
</table>
```

本实例不但应用类中的方法对字符串进行了截取，而且还使用 substr()函数对字符串进行了截取，与类中的方法进行对比，运行结果如图 14.13 所示。



图 14.13　通过类中的方法截取中文字符串

14.5　小　　结

本章主要介绍了面向对象的概念、特点和 PHP 5 中的新特性，如抽象类、接口、克隆等。虽然本章关于 OOP 概念介绍得很全面、很详细，但要想真正明白面向对象思想，必须要多动手实践、多动脑思考、注意平时积累等。希望读者通过自己的努力，能有所突破。

14.6　练习与实践

1．PHP 显示中文时，经常会出现乱码，编写一个编码转换类，从而实现编码的自动转换。**（答案位置：光盘\TM\sl\14\25）**

2．做 Web 开发时，需要对各种情况作出处理，并输出相应的信息。编写一个输出类，根据不同的情况，输出不同的处理结果。**（答案位置：光盘\TM\sl\14\26）**

第 15 章

PHP 加密技术

（视频讲解：31 分钟）

随着网络的普及，网上购物已经成了时尚男女的主要消费方式之一，因此对于个人隐私、账号密码等敏感数据的保护也越来越重要。原来只有在侦探小说中才听说过的加密、解密也出现在现实生活中。其实加密技术本没有那么神秘，它就是一种相对比较复杂的算法。作为普通的开发者来说，可以使用一些已有的、比较有名气的加密算法，如使用 MD5、SHA 等自己创建加密函数。

通过阅读本章，您可以：

- 熟练掌握 PHP 内置加密函数的使用方法
- 熟练掌握单向加密函数的使用方法
- 熟练掌握 PHP 加密扩展库的使用方法
- 熟练掌握双向加密函数的使用方法
- 熟练掌握 MD5 校验码的使用方法

15.1 PHP 加密函数



视频讲解：光盘\TM\lx\15\PHP 加密函数.exe

数据加密的基本原理就是对原来为明文的文件或数据按某种算法进行处理，使其成为不可读的一段代码，通常称为“密文”，通过这样的途径来达到保护数据不被非法窃取和阅读的目的。

在 PHP 中能对数据进行加密的函数主要有 crypt()、md5()和 sha1()，还有加密扩展库 Mcrypt 和 Mash。这里主要介绍其中的 3 种：crypt()函数、md5()函数和 sha1()函数。

15.1.1 使用 crypt()函数进行加密

crypt()函数可以完成单向加密功能，语法如下：

```
string crypt(string str[, string salt]);
```

其中，str 参数是需要加密的字符串，salt 参数为加密时使用的干扰串。如果省略掉第二个参数，则会随机生成一个干扰串。crypt()函数支持 4 种算法和长度，如表 15.1 所示。

表 15.1 crypt()函数支持的 4 种算法和 salt 参数的长度

算　法	salt 长度
CRYPT_STD_DES	2-character（默认）
CRYPT_EXT_DES	9-character
CRYPT_MD5	12-character（以1开头）
CRYPT_BLOWFISH	16-character（以2开头）

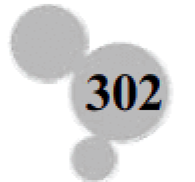
说明

默认情况下，PHP 使用一个或两个字符的 DES 干扰串，如果系统使用的是 MD5，则会使用 12 个字符。可以通过 CRYPT_SALT_LENGTH 变量来查看当前所使用的干扰串的长度。

【例 15.1】 首先声明一个字符串变量$str，赋值为“This is an example!”，然后使用 crypt()函数进行加密并输出。实例代码如下：**（实例位置：光盘\TM\sl\15\1）**

```
<?php
    $str = 'This is an example!';                        //声明字符串变量$str
    echo '加密前$str 的值为：'.$str;
    $crypttostr = crypt($str);                           //对变量$str 加密
    echo '<p>加密后$str 的值为：'.$crypttostr;           //输出加密后的变量
?>
```

运行结果如图 15.1 所示。



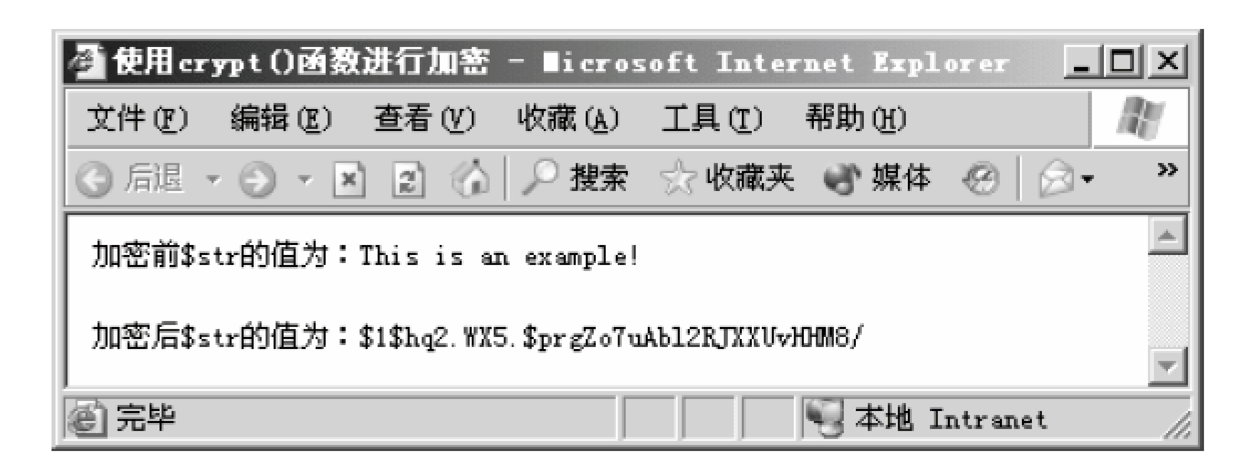


图 15.1　使用 crypt()函数进行加密

按 F5 键刷新，会发现每次生成的加密结果都不相同，那么该如何对加密后的数据进行判断就成为了问题，crypt()函数是单向加密的，密文不可还原成明码，而每次加密后的数据还不相同，这就是 salt 参数要解决的问题。crypt()函数用 salt 参数对明文进行加密，判断时，对输出的信息再次使用相同的 salt 参数进行加密，对比两次加密后的结果来进行判断。

【例 15.2】 本例对输入的用户名进行检测，如果该用户存在，显示“用户名已存在。”；否则显示“恭喜您：用户名可以使用!”。实例代码如下：**（实例位置：光盘\TM\sl\15\2）**

```
<?php
    /*    连接数据库    */
    $conn = mysql_connect("localhost","root","root") or die("数据库连接错误".mysql_error());
    mysql_select_db("db_database15",$conn) or die("数据库访问错误".mysql_error());
    mysql_query("set names gb2312");
?>
<form id="form1" name="form1" method="post" action="">
    <td><input name="username" type="text" id="username" size="15" /></td>
    <td><input type="submit" name="Submit" value="检查" id="Submit" /></td>
</form>
<?php
    if(trim($_POST[username]) != ""){                          //trim()函数去掉字符串两边的空格
        $usr = crypt(trim($_POST[username]),"tm");             //对用户名进行加密
        $sql = "select * from tb_user where user = '".$usr."'";     //生成查询语句
        $rst = mysql_query($sql,$conn);                        //执行语句，返回结果集
        if(mysql_num_rows($rst) > 0){                          //如果结果集大于 0
            echo "<font color='red'>用户名已存在。</font>";      //说明用户名存在
        }else{                                                 //否则说明该用户名可用
            echo "<font color='green'>恭喜您：用户名可以使用!</font>";
        }
    }
?>
```

运行结果如图 15.2 所示。

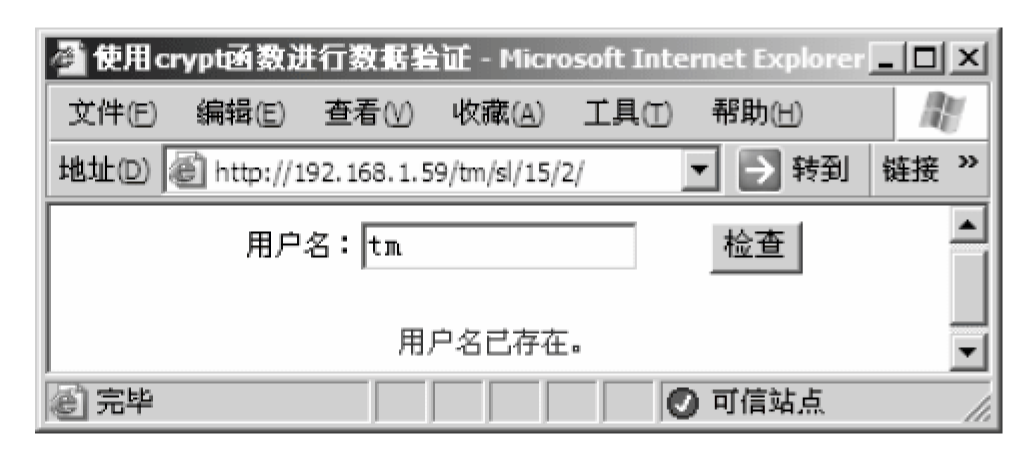


图 15.2　使用 crypt()函数进行数据验证

注意

实例代码中加粗显示的函数为数据库操作函数，如果读者对 PHP 链接 MySQL 数据库不了解，可以先参考第 16 章 MySQL 数据库基础，然后再回来学习本实例。

15.1.2 使用 md5()函数进行加密

md5()函数使用 MD5 算法。MD5 的全称是 Message-Digest Algorithm 5（信息-摘要算法），它的作用是把不同长度的数据信息经过一系列的算法计算成一个 128 位的数值，就是把一个任意长度的字节串变换成一定长的大整数。注意这里是“字节串”而不是“字符串”，因为这种变换只与字节的值有关，与字符集或编码方式无关。md5()函数的格式如下：

```
string md5 ( string str [, bool raw_output] );
```

其中字符串 str 为要加密的明文，raw_output 参数如果设为 true，则函数返回一个二进制形式的密文，该参数默认为 false。

很多网站注册用户的密码都是先使用 MD5 加密，然后再保存到数据库中的。用户登录时，程序把用户输入的密码计算成 MD5 值，然后再去和数据库中保存的 MD5 值进行比较。在这个过程中，程序自身都不会“知道”用户的真实密码，从而保证注册用户的个人隐私，提高安全性。

【例 15.3】 本例实现会员注册和登录的功能，将会员注册的密码通过 md5()进行加密后，然后保存到数据库中。其操作步骤如下：**（实例位置：光盘\TM\sl\15\3）**

（1）创建 conn.php 文件，完成与 db_database15 数据库的连接。其代码如下：

```
<?php
    $conn=mysql_connect("localhost","root","root") or die("数据库连接失败".mysql_error());    //连接服务器
    mysql_select_db("db_database15",$conn);            //连接数据库
    mysql_query("set names gb2312");                   //设置编码格式
?>
```

（2）创建会员注册页面，即 register.php 文件。在该文件中，创建 form 表单，通过 register()方法对表单元素值进行验证；添加表单元素，完成会员名和密码的提交；将表单中数据提交到 register_ok.php 文件中，通过面向对象的方法完成会员注册信息的提交操作。其注册页面如图 15.3 所示。



图 15.3　会员注册页面

（3）创建 register_ok.php 文件，获取表单中提交的数据，通过 md5()函数对密码进行加密，使用

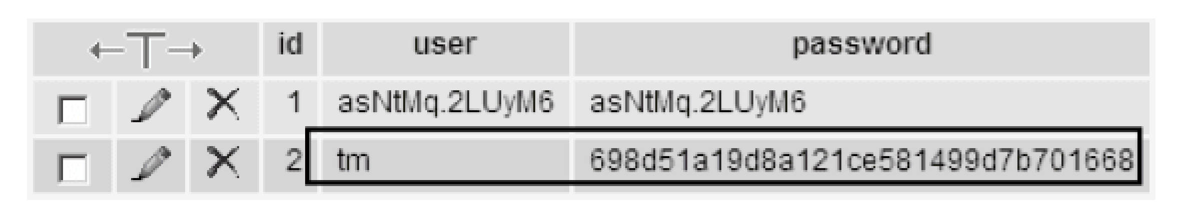


面向对象的方法完成会员注册信息的提交。其代码如下：

```
<?php
class chkinput {                                    //定义 chkinput 类
    var $name;                                      //定义成员变量
    var $pwd;                                       //定义成员变量
    function chkinput($x, $y) {                     //定义成员方法
        $this->name = $x;                           //为成员变量赋值
        $this->pwd = $y;                            //为成员变量赋值
    }
    function checkinput() {                         //定义方法，完成用户注册
        include "conn/conn.php";                    //通过 include 调用数据库连接文件
        $info = mysql_query ( "insert into tb_user(user,password)value('" . $this->name . "','" . $this->pwd . "')" );
        if ($info == false) {                       //根据添加操作的返回结果，给出提示信息
            echo "<script language='javascript'>alert('会员注册失败！');history.back();</script>";
            exit ();
        } else {
            $_SESSION [admin_name] = $this->name;       //注册成功后，将用户名赋给 SESSION 变量
            echo "<script language='javascript'>alert('恭喜您，注册成功！');window.location.href='index.php';
</script>";
        }
    }
}
$obj = new chkinput ( trim ( $_POST [name] ), trim ( md5 ( $_POST [pwd] ) ) );        //实例化类
$obj->checkinput ();                                //根据返回对象调用方法执行注册操作
?>
```

（4）创建 index.php 和 index_ok.php 文件，实现会员登录的功能，具体代码可参考光盘中的内容，这里不作讲解。

在会员注册成功后，可以查看一下存储在数据库中的数据，通过 MD5 加密后的密码如图 15.4 所示。



←T→	id	user	password
	1	asNtMq.2LUyM6	asNtMq.2LUyM6
	2	tm	698d51a19d8a121ce581499d7b701668

图 15.4　MD5 加密后的密码

15.1.3　使用 sha1()函数进行加密

和 MD5 类似的还有 SHA 算法。SHA 全称为 Secure Hash Algorithm（安全哈希算法），PHP 提供的 sha1()函数使用的就是 SHA 算法，函数的语法如下：

```
string sha1 ( string str [, bool raw_output] )
```

函数返回一个 40 位的十六进制数，如果参数 raw_output 为 true，则返回一个 20 位的二进制数。默认 raw_output 为 false。



注意

sha 后面的 1 是阿拉伯数字（1、2、3）里的 1，不是字母 l（L），读者一定要注意。

【例 15.4】本例对一字符串进行 MD5 和 SHA 加密运算，实例代码如下：（**实例位置：光盘\TM\sl\15\4**）

```
<?
php echo md5('PHPER');                    //使用 md5()函数加密字符串 PHPER
php echo sha1('PHPER');                   //使用 sha1()函数加密字符串 PHPER
?>
```

MD5 加密运算和 SHA 加密运算字符串的对比效果如图 15.5 所示。

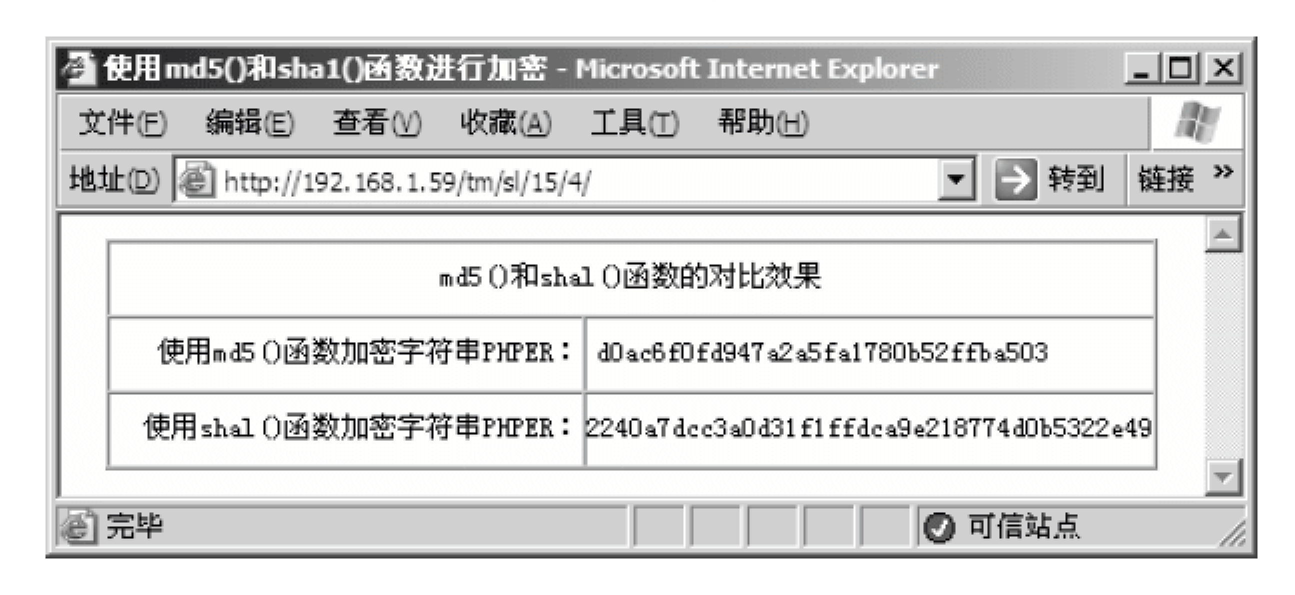


图 15.5　使用 md5()和 sha1()函数的效果对比

15.2　PHP 加密扩展库

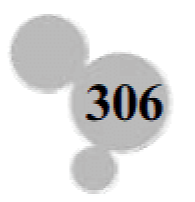
视频讲解：光盘\TM\lx\15\PHP 加密扩展库.exe

PHP 除了自带的几种加密函数外，还有功能更全面的加密扩展库 Mcrypt 和 Mhash。其中 Mcrypt 扩展库可以实现加密解密功能，既能将明文加密，也可以将密文还原。Mhash 扩展库则包含了 MD5 在内的多种 hash 算法实现的混编函数。

15.2.1　Mcrypt 扩展库

1. Mcrypt 库安装

单向加密的优势是密文无法还原为明文，即使数据被截获也不会造成资料外泄。但有时，还需要将密文还原成明文，这时就需要使用双向加密技术了。Mcrypt 是一个功能十分强大的加密算法扩展库。在标准的 PHP 安装过程中并没有安装 Mcrypt，但 PHP 的主目录下包含了 libmcrypt.dll 和 libmhash.dll 文件（libmhash.dll 是 Mhash 扩展库，这里一起安装上），首先将文件复制到系统目录 windows\system32 下，然后在 php.ini 文件中找到“;extension=php_mcrypt.dll”和“;extension=php_mhash.dll”这两个语句，将前面的分号“;”去掉，最后重新启动服务器，即可使用这两个扩展库。



2. Mcrypt 库常量

【例 15.5】Mcrypt库支持20多种加密算法和8种加密模式，读者可以通过函数mcrypt_list_ algorithms()和 mcrypt_list_modes()查看，实例代码如下：（**实例位置：光盘\TM\sl\15\5**）

```
<?php
	$en_dir = mcrypt_list_algorithms();				//函数返回 Mcrypt 支持的加密算法数组
	echo "Mcrypt 支持的算法有：";
	foreach($en_dir as $en_value){
		echo $en_value." ";
	}
	$mo_dir = mcrypt_list_modes();					//函数返回 Mcrypt 支持的算法模式数组
	echo "<p>Mcrypt 支持的加密模式有：";
	foreach($mo_dir as $mo_value){
		echo $mo_value." ";
	}
?>
```

运行结果如图 15.6 所示。

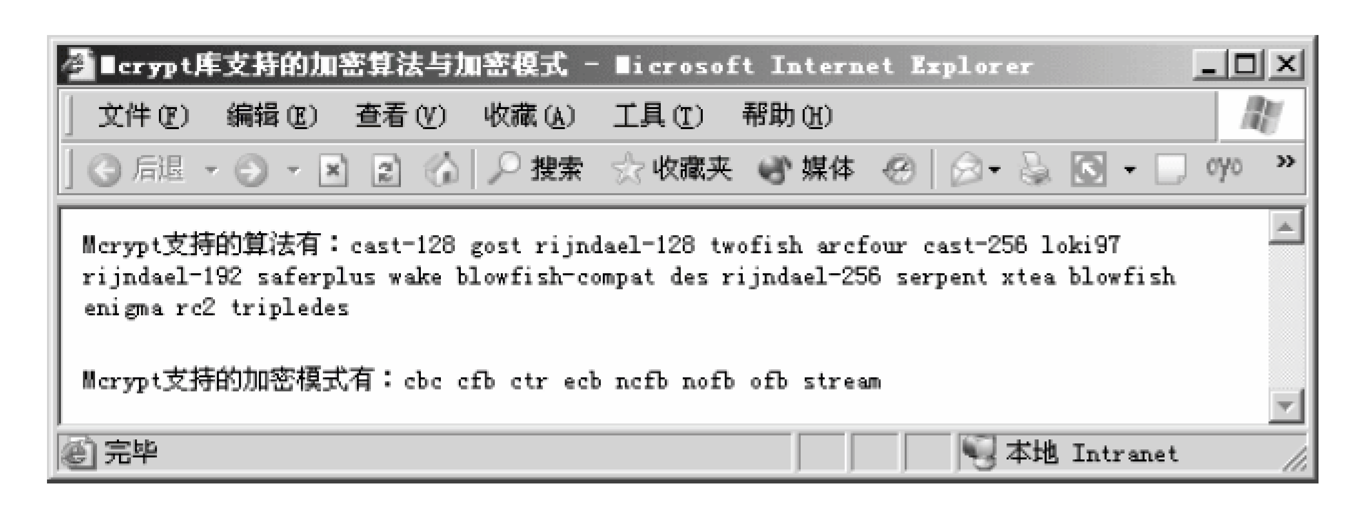


图 15.6　Mcrypt 库支持的加密算法与加密模式

这些算法和模式在实际应用中要用常量来表示，写的时候分别加上前缀 MCRYPT_和 MCRYPT_MODE_来表示，如：

- ☑ TWOFISH 算法表示为 MCRYPT_TWOFISH。
- ☑ CBC 加密模式表示为 MCRYPT_MODE_CBC。

3. Mcrypt 应用

【例 15.6】 使用 Mcrypt 进行加密和解密不像使用 md5()、sha1()等函数，直接调用即可。为了让读者能清楚地了解 Mcrypt 的工作流程，下面通过一个实例来学习 Mcrypt 是如何工作的。实例代码如下：（**实例位置：光盘\TM\sl\15\6**）

```
<?php
	$str = "被加密的内容：相见时难别亦难  东风无力百花残";			//加密文本
	$key = "key:111";										//密钥
	$cipher = MCRYPT_DES;									//密码类型
	$modes = MCRYPT_MODE_ECB;								//密码模式
	$iv = mcrypt_create_iv(mcrypt_get_iv_size($cipher,$modes),MCRYPT_RAND);	//初始化向量
	echo "加密前：".$str."<p>";
	//加密：
	$str_encrypt = mcrypt_encrypt($cipher,$key,$str,$modes,$iv);		//加密函数
```

```
    echo "加密后：".$str_encrypt." <p>";
    $str_decrypt = mcrypt_decrypt($cipher,$key,$str_encrypt,$modes,$iv);        //解密函数
    echo "还原：".$str_decrypt."<p>";
?>
```

运行结果如图 15.7 所示。

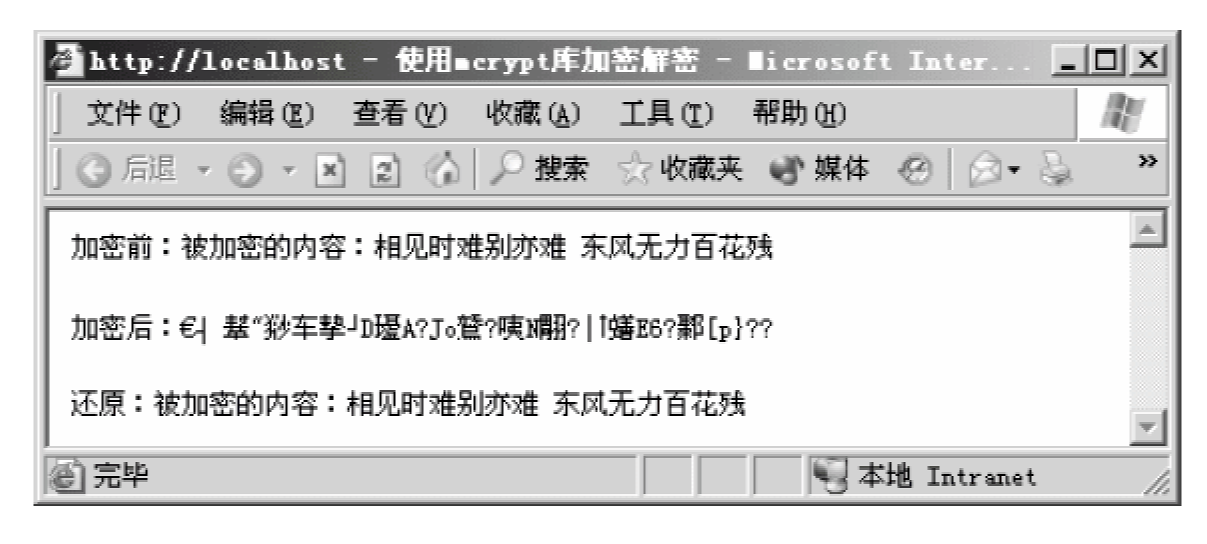


图 15.7　使用 Mcrypt 库加密解密

下面对代码中加粗的函数进行讲解。

（1）string mcrypt_create_iv (int size [, int source])

使用 Mcrypt 进行数据加密、解密之前，首先要创建一个初始化向量（简称 iv）。创建初始化向量需要两个参数：size 指定了 iv 的大小，source 为 iv 的源。source 可以取如下值。

☑ MCRYPT_RAND：系统随机数。

☑ MCRYPE_DEV_RANDOM：读取目录/dev/random 中的数据（UNIX 系统）。

☑ MCRYPT_DEV_URANDOM：读取目录/dev/urandom 中的数据（UNIX 系统）。

（2）nt mcrypt_get_iv_size (string cipher, string mode)

该函数返回初始化向量（iv）的大小。函数中的两个参数是前面刚介绍过的加密算法（cipher）和算法模式（mode）。

（3）string mcrypt_encrypt (string cipher, string key, string data, string mode [, string iv])

初始化向量后，即可使用 mcrypt_encrypt()加密函数对数据进行加密。该函数的 5 个参数分别如下。

☑ cipher：加密算法。例 15.6 中为变量$cipher，这里的加密算法可以和初始化向量中的加密算法不一样。

☑ key：密钥。例 15.6 中的变量$key。

☑ data：需要加密的数据。例 15.6 中的变量$str。

☑ mode：算法模式。例 15.6 中的变量$modes，可以和初始化向量中的模式不一样。

☑ iv：初始化向量。例 15.6 中的变量$iv。

（4）string mcrypt_decrypt (string cipher, string key, string data, string mode [, string iv])

解密函数 mcrypt_decrypt()和加密函数 mcrypt_encrypt()的参数几乎是一样的。唯一不同的是参数 data，这里的 data 为需要解密的数据，而不是原始数据。在例 15.6 中为需要解密的变量$str_encrypt。

注意

加密函数和解密函数中的 cipher、key 和 mode 参数必须要一致，否则数据不会被还原。

Mcrypt 扩展库包含 30 多个函数，对加密技术感兴趣的读者可以参考 PHP 手册，其中有详细的介绍。



15.2.2　Mhash 扩展库

1．Mhash 库安装

关于 Mhash 库的安装在前面已经介绍过，这里不再重复。读者可以参见 15.2.1 节中的 Mcrypt 库安装。

2．Mhash 库常量

【例 15.7】 Mhash 库支持 MD5、SHA、CRC32 等多种散列算法，可以使用 mhash_count()和 mhash_get_hash_name()函数输出支持的算法名称。实例代码如下：**（实例位置：光盘\TM\sl\15\7）**

```
<?php
$num = mhash_count();                                   //函数返回最大的 hash id
echo "Mhash 库支持的算法有：";
for($i = 0; $i <= $num; $i++){
    echo $i."=>".mhash_get_hash_name($i)."   ";         //输出每一个 hash id 的名称
}
?>
```

运行结果如图 15.8 所示。



图 15.8　Mhash 库支持的加密算法

如果在实际应用中使用上面的常量，需要在算法名称前面加上 MHASH_前缀，如 CRC32 表示为 MHASH_CRC32。

3．Mhash 应用

相比 Mcrypt 扩展库的 30 多个函数，Mhash 库中只有 5 个函数，除了上面使用到的两个函数外，下面来介绍其他的 3 个函数。

☑　mhash_get_block_size()函数

函数语法：

```
int mhash_get_block_size ( int hash )
```

该函数用来获取参数 hash 的区块大小，如 mhash_get_block_size(MHASH_CRC32)。

☑　mhash()函数

函数语法：

```
string mhash ( int hash, string data [, string key] )
```

该函数返回一个哈希值。参数 hash 为要使用的算法，参数 data 是要加密的数据，参数 key 是加密

使用的密钥。

☑ mhash_keygen_s2k()函数

函数语法：

```
string mhash_keygen_s2k ( int hash, string password, string salt, int bytes )
```

该函数将根据参数 password 和 salt 返回一个长度为字节的 key 值，参数 hash 为要使用的算法。其中 salt 为一个固定 8 字节的值，如果用户给出的数值小于 8 字节，将用 0 补齐。

【例 15.8】 下面使用 mash()函数和 mhash_keygen_s2k()函数生成一个校验码，并使用 bin2hex()函数将二进制结果转换为十六进制。实例代码如下：**（实例位置：光盘\TM\sl\15\8）**

```
<?php
    $filename = '08.txt';                                  //文件路径
    $str = file_get_contents($filename);                   //读取文件内容到变量$str 中
    $hash = 2;                                             //设置 hash 值
    $password = '111';                                     //设置变量$password
    $salt = '1234';                                        //设置变量$salt
    $key = mhash_keygen_s2k(1,$password,$salt,10);         //生成 key 值
    $str_mhash = bin2hex(mhash($hash,$str,$key));          //使用$key 值、$hash 值对字串$str 加密
    echo "文件 08.txt 的校验码是：".$str_mhash;              //输出校验码
?>
```

运行结果如图 15.9 所示。



图 15.9　使用 Mhash 库生成校验码

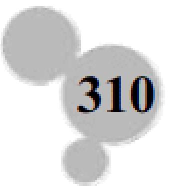
15.3　小　　结

本章中首先介绍了 PHP 中的加密函数 crypt()、md5()和 sha1()，又学习了 PHP 扩展库 Mcrypt 和 Mhash。其中，属于单向加密的有 crypt()、md5()、sha1()和 Mhash 扩展库，可以还原密文的是 Mcrypt 扩展库。相信通过本章的学习及实践，读者可以熟练地使用各种加密手段对敏感数据进行保护。

15.4　练习与实践

1．分别使用 crypt()和 md5()函数做一个用户登录验证页，以验证用户登录所使用的用户名和密码是否正确。**（答案位置：光盘\TM\sl\15\9）**

2．使用 OR、XOR 等运算符，自定义一个加密函数。**（答案位置：光盘\TM\sl\15\10）**



第16章

MySQL 数据库基础

（视频讲解：57 分钟）

只有与 MySQL 数据库相结合，才能充分发挥动态网页编程语言的魅力，因为网络上的众多应用都是基于数据库的。PHP 支持多种数据库，尤其与 MySQL 被称为黄金组合，本章就带领读者来学习 MySQL。MySQL 命令行通过 sql 语句对数据库进行操作，本章将详细介绍 MySQL 数据库的基础知识，通过本章的学习，读者不但可以轻松掌握操作 MySQL 数据库、数据表的方法，还可以学习到对 MySQL 数据库进行查询等操作。

通过阅读本章，您可以：

- 了解 MySQL 是什么、能做什么
- 了解 MySQL 的特点
- 掌握启动、连接、断开和停止 MySQL 服务器的方法
- 掌握操作 MySQL 数据库的技术
- 掌握操作 MySQL 数据表的技术
- 掌握操作 MySQL 语句的技术
- 掌握 MySQL 数据库备份和恢复的方法

16.1　MySQL 概述



Michael Monty Widenius

MySQL 是目前最为流行的开源的数据库，是完全网络化的跨平台关系型数据库系统，它是由瑞典的 MySQL AB 公司开发的，由 MySQL 的初始开发人员 David Axmark 和 Michael Monty Widenius（见右图）于 1995 年建立。它的象征符号是一只名为 Sakila 的海豚，代表着 MySQL 数据库和团队的速度、能力、精确和优秀本质。

MySQL 数据库可以称得上是目前运行速度最快的 SQL 语言数据库。除了具有许多其他数据库所不具备的功能和选择之外，MySQL 数据库还是一种完全免费的产品，用户可以直接从网上下载使用，而不必支付任何费用。

下面介绍 MySQL 的特点。

☑ 功能强大：MySQL 中提供了多种数据库存储引擎，各个引擎各有所长，适用于不同的应用场合，用户可以选择最合适的引擎以得到最高性能，甚至可以处理每天访问量数亿的高强度 Web 搜索站点。MySQL 支持事务、视图、存储过程和触发器等。

☑ 支持跨平台：MySQL 支持至少 20 种以上的开发平台，包括 Linux、Windows、FreeBSD、IBMAIX、AIX 和 FreeBSD 等。这使得在任何平台下编写的程序都可以进行移植，而不需要对程序做任何修改。

☑ 运行速度快：高速是 MySQL 的显著特性。在 MySQL 中，使用了极快的 B 树磁盘表（MyISAM）和索引压缩；通过使用优化的单扫描多连接，能够极快地实现连接；SQL 函数使用高度优化的类库实现，运行速度极快。

☑ 支持面向对象：PHP 支持混合编程方式。编程方式可分为纯粹面向对象、纯粹面向过程、面向对象与面向过程混合 3 种方式。

☑ 安全性高：灵活安全的权限和密码系统允许基本主机的验证。连接到服务器时，所有的密码传输均采用加密形式，从而保证了密码的安全。

☑ 成本低：MySQL 数据库是一种完全免费的产品，用户可以直接从网上下载。

☑ 支持各种开发语言：MySQL 为各种流行的程序设计语言提供支持，为它们提供了很多的 API 函数，包括 PHP、ASP.NET、Java、Eiffel、Python、Ruby、Tcl、C、C++和 Perl 等。

☑ 数据库存储容量大：MySQL 数据库的最大有效表尺寸通常是由操作系统对文件大小的限制决定的，而不是由 MySQL 内部限制决定的。InnoDB 存储引擎将 InnoDB 表保存在一个表空间内，该表空间可由数个文件创建，表空间的最大容量为 64TB，可以轻松处理拥有上千万条记录的大型数据库。

☑ 支持强大的内置函数：PHP 中提供了大量内置函数，几乎涵盖了 Web 应用开发中的所有功能。它内置了数据库连接、文件上传等功能，MySQL 支持大量的扩展库，如 MySQLi 等，为快速开发 Web 应用提供方便。

16.2　启动、连接、断开和停止 MySQL 服务器

视频讲解：光盘\TM\lx\16\启动、连接、断开和停止 MySQL 服务器.exe

通过系统服务器和命令提示符（DOS）均可启动、连接和关闭 MySQL，操作非常简单。但通常情况下，建议不要停止 MySQL 服务器，否则数据库将无法使用。

16.2.1　启动 MySQL 服务器

启动 MySQL 服务器的方法有两种：系统服务器和命令提示符（DOS）。下面以 Windows 2003 Server 为例具体介绍每种方法的操作流程。

1．通过系统服务器启动 MySQL 服务器

如果 MySQL 设置为 Windows 服务，则可以通过选择“开始”/“管理工具”/“服务”命令打开 Windows 服务管理器。在服务器的列表中找到 mysql 服务并右击，在弹出的快捷菜单中选择“启动”命令，启动 MySQL 服务器，如图 16.1 所示。



图 16.1　通过系统服务启动 MySQL 服务器

2．在命令提示符下启动 MySQL 服务器

选择“开始”/“运行”命令，在弹出的“运行”对话框中输入 cmd 命令，按 Enter 键进入 DOS 窗口。在命令提示符下输入：

```
\> net start mysql
```

按 Enter 键，即可启用 MySQL 服务器，如图 16.2 所示。

```
Microsoft Windows [版本 5.2.3790]
(C) 版权所有 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>net start mysql
mysql 服务正在启动 .
mysql 服务已经启动成功。
```

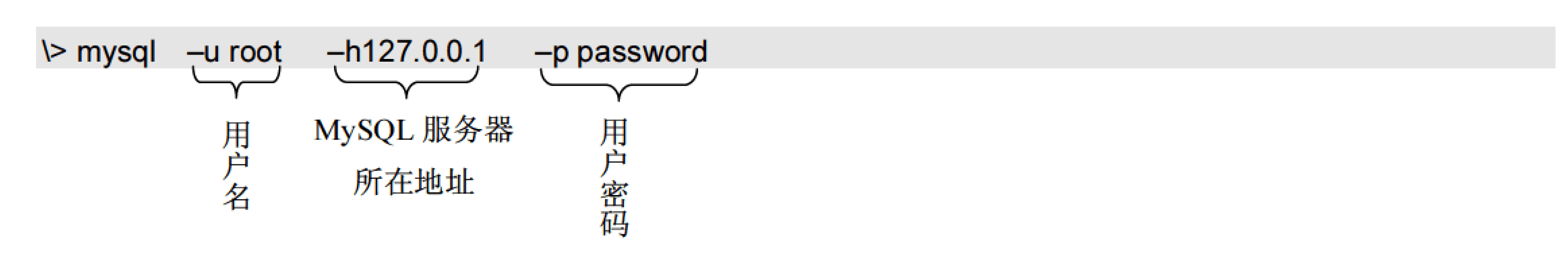
图 16.2　在命令提示符下启动 MySQL 服务器

16.2.2　连接和断开 MySQL 服务器

下面分别介绍连接和断开 MySQL 服务器的方法。

1．连接 MySQL 服务器

连接 MySQL 服务器通过 mysql 命令实现。在 MySQL 服务器启动后，选择“开始”/“运行”命令，在弹出的“运行”对话框中输入 cmd 命令，按 Enter 键后进入 DOS 窗口，在命令提示符下输入：



注意

在连接 MySQL 服务器时，MySQL 服务器所在地址（如–h127.0.0.1）可以省略不写。

输入命令语句后，按 Enter 键即可连接 MySQL 服务器，如图 16.3 所示。

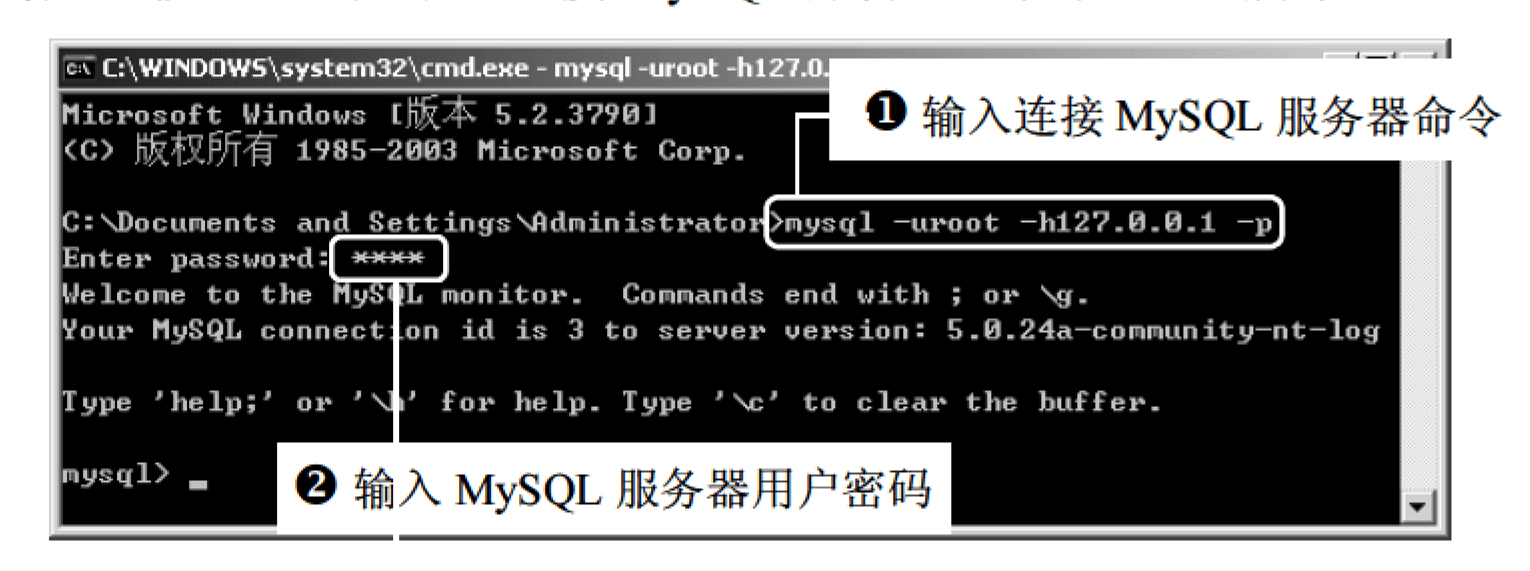


图 16.3　连接 MySQL 服务器

技巧

为了保护 MySQL 数据库的密码，可以采用如图 16.3 所示的密码输入方式。如果密码在-p 后直接给出，那么密码就以明文显示，例如：

mysql –u root –h127.0.0.1 –p **root**

按 Enter 键后再输入密码将以加密的方式显示，然后按 Enter 键即可成功连接 MySQL 服务器。



如果用户在使用 mysql 命令连接 MySQL 服务器时弹出如图 16.4 所示的信息，说明用户未设置系统的环境变量。

根据图 16.4 弹出的错误提示进行分析，mysql 文件位于 MySQL 安装目录的 bin 文件夹下，如果没有将 bin 文件加入到 Windows 环境变量 PATH 中，则需要使用 mysql 文件的绝对路径。

在通常情况下，如果用户采用独立安装方式安装 MySQL 数据库，那么需要设置系统的环境变量；如果采用 AppServ 集成化安装包安装 MySQL 数据库，系统会自动设置环境变量，不需要用户手动设置。下面介绍环境变量的设置方法。

右击“我的电脑”，在弹出的快捷菜单中选择“属性”命令，弹出“系统属性”对话框，系统环境变量的设置如图 16.5 所示。



图 16.4　连接 MySQL 服务器出错

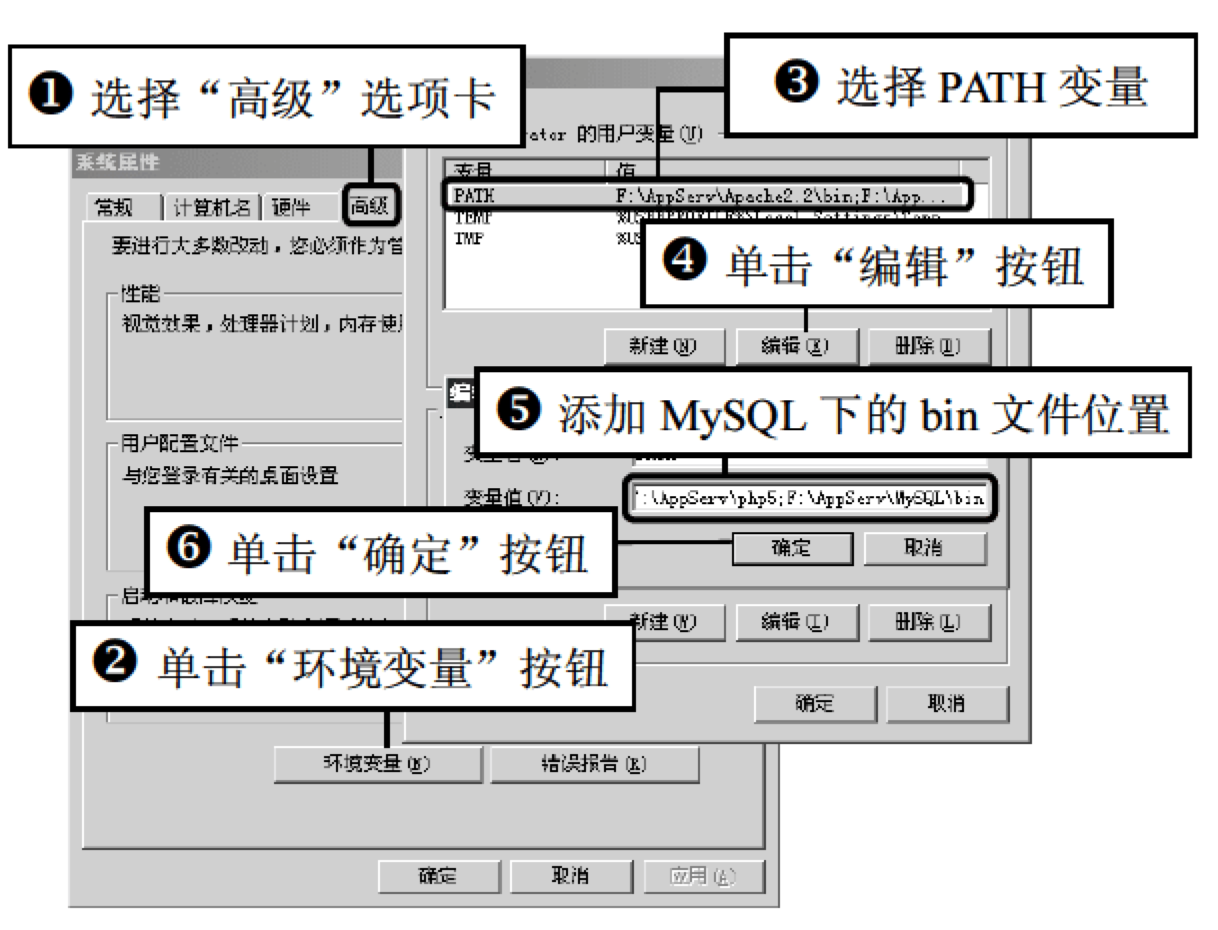


图 16.5　设置系统的环境变量

环境变量设置完成后，再使用 mysql 命令即可成功连接 MySQL 服务器。

2. 断开 MySQL 服务器

连接到 MySQL 服务器后，可以通过在 MySQL 提示符下输入 exit 或者 quit 命令断开 MySQL 连接，格式如下：

```
mysql> quit;
```

16.2.3　停止 MySQL 服务器

停止 MySQL 服务器的方法有 3 种：系统服务器、命令提示符（DOS）和 mysqladmin 命令，下面具体介绍每种方法的操作流程。

1. 通过系统服务器停止 MySQL 服务器

如果将 MySQL 设置为 Windows 服务，则可以通过选择“开始”/“管理工具”/“服务”命令，打开 Windows 服务管理器，在服务器的列表中右击 mysql 服务，在弹出的快捷菜单中选择“停止”命令，

停止 mysql 服务，如图 16.6 所示。



图 16.6　停止 MySQL 服务器

2. 在命令提示符下停止 MySQL 服务器

选择“开始”/“运行”命令，在弹出的“运行”对话框中输入 cmd 命令，进入 DOS 窗口，在命令提示符下输入：

```
\> net stop mysql
```

按 Enter 键即可停止 MySQL 服务器，如图 16.7 所示。

3. 使用 mysqladmin 命令停止 MySQL 服务器

选择“开始”/“运行”命令，在弹出的“运行”对话框中输入 cmd 命令，进入 DOS 窗口，在命令提示符下输入：

```
C:\Documents and Settings\Administrator> mysqladmin –uroot shutdown –proot
```

按 Enter 键即可停止 MySQL 服务，如图 16.8 所示。



图 16.7　在命令提示符下停止 MySQL 服务器

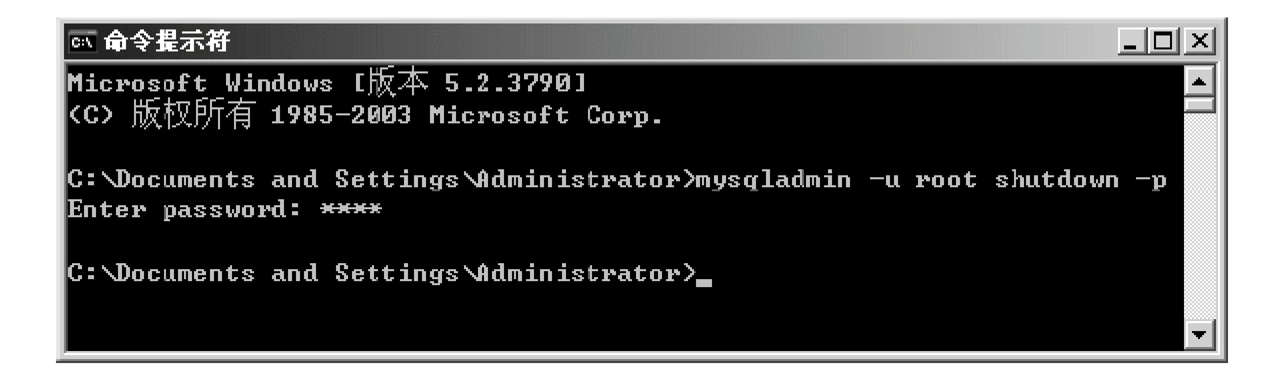


图 16.8　使用 mysqladmin 命令停止 MySQL 服务器

16.3　MySQL 数据库操作

视频讲解：光盘\TM\lx\16\MySQL 数据库操作.exe

启动并连接 MySQL 服务器后，即可对 MySQL 数据库进行操作，操作 MySQL 数据库的方法非常简单，下面进行详细介绍。



16.3.1　创建数据库 CREATE DATABASE

使用 CREATE DATABASE 语句可以轻松创建 MySQL 数据库。

语法格式如下：

```
CREATE　DATABASE　数据库名;
```

在创建数据库时，数据库命名有以下几项规则：

☑　不能与其他数据库重名，否则将发生错误。

☑　名称可以由任意字母、阿拉伯数字、下划线（_）和“$”组成，可以使用上述的任意字符开头，但不能使用单独的数字，否则会造成它与数值相混淆。

☑　名称最长可为 64 个字符，而别名最多可长达 256 个字符。

☑　不能使用 MySQL 关键字作为数据库名、表名。

☑　在默认情况下，Windows 下数据库名、表名的大小写是不敏感的，而在 Linux 下数据库名、表名的大小写是敏感的。为了便于数据库在平台间进行移植，建议采用小写来定义数据库名和表名。

例如，通过 CREATE DATABASE 语句创建一个名称为 db_admin 的数据库，如图 16.9 所示。



图 16.9　创建 MySQL 数据库

16.3.2　查看数据库 SHOW DATABASES

成功创建数据库后，可以使用 SHOW 命令查看 MySQL 服务器中的所有数据库信息。

语法格式如下：

```
SHOW　DATABASES;
```

例如，在 16.3.1 节中创建了数据库 db_admin，下面使用 SHOW DATABASES 语句查看 MySQL 服务器中的所有数据库名称，如图 16.10 所示。



图 16.10　查看数据库

从图 16.10 运行的结果可以看出，通过 SHOW 命令查看 MySQL 服务器中的所有数据库，结果显示 MySQL 服务器中有 4 个数据库。

16.3.3　选择数据库 USE DATABASE

在上面的讲解中，虽然成功创建了数据库，但并不表示当前就在操作数据库 db_admin。可以使用 USE 语句选择一个数据库，使其成为当前默认数据库。

语法格式如下：

```
USE　数据库名;
```

例如，选择名称为 db_admin 的数据库，设置其为当前默认的数据库，如图 16.11 所示。

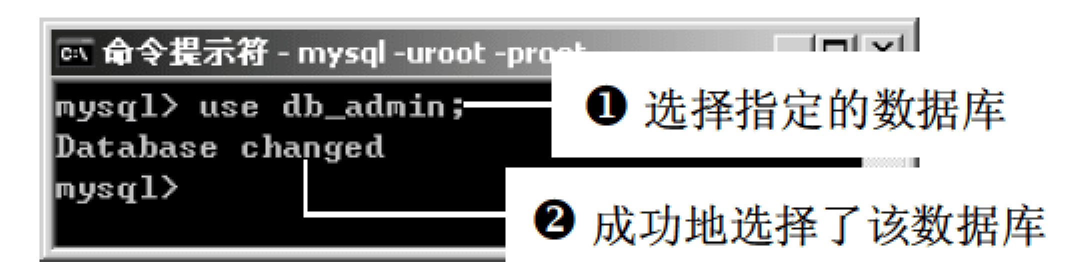


图 16.11　选择数据库



当用户成功选择数据库后，即可使用 sql 语句针对该数据库进行操作。

16.3.4　删除数据库 DROP DATABASE

删除数据库可以使用 DROP DATABASE 语句。

语法格式如下：

```
DROP DATABASE　数据库名;
```

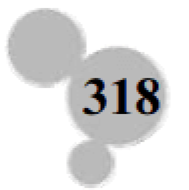
注意

删除数据库的操作应该谨慎使用，一旦执行该操作，数据库的所有结构和数据都会被删除，没有恢复的可能，除非数据库有备份。

例如，通过 DROP DATABASE 语句删除名称为 db_admin 的数据库，如图 16.12 所示。



图 16.12　删除数据库



16.4　MySQL 数据表操作

视频讲解：光盘\TM\lx\16\MySQL 数据表操作.exe

在对 MySQL 数据表进行操作之前，必须首先使用 USE 语句选择数据库，才可在指定的数据库中对数据表进行操作，如创建数据表、修改表结构、数据表更名或删除数据表等，否则是无法对数据表进行操作的。下面介绍对数据表的操作方法。

16.4.1　创建数据表 CREATE TABLE

创建数据表可以使用 CREATE TABLE 语句。

语法格式如下：

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] 数据表名
[(create_definition,…)][table_options] [select_statement]
```

CREATE TABLE 语句的参数说明如表 16.1 所示。

表 16.1　CREATE TABLE 语句的参数说明

关　键　字	说　　明
TEMPORARY	如果使用该关键字，表示创建一个临时表
IF NOT EXISTS	该关键字用于避免表不存在时 MySQL 报告的错误
create_definition	表的列属性部分。MySQL 要求在创建表时，表至少包含一列
table_options	表的一些特性参数
select_statement	SELECT 语句描述部分，用它可以快速地创建表

下面介绍列属性 create_definition 部分，每一列定义的具体格式如下：

```
col_name   type [NOT NULL | NULL] [DEFAULT default_value] [AUTO_INCREMENT]
           [PRIMARY KEY ] [reference_definition]
```

属性 create_definition 的参数说明如表 16.2 所示。

表 16.2　属性 create_definition 的参数说明

参　　数	说　　明
col_name	字段名
type	字段类型
NOT NULL | NULL	指出该列是否允许是空值，系统一般默认允许为空值，所以当不允许为空值时，必须使用 NOT NULL
DEFAULT default_value	表示默认值
AUTO_INCREMENT	表示是否是自动编号，每个表只能有一个 AUTO_INCREMENT 列，并且必须被索引

续表

参　数	说　明
PRIMARY KEY	表示是否为主键。一个表只能有一个 PRIMARY KEY。如表中没有一个 PRIMARY KEY，而某些应用程序需要 PRIMARY KEY，MySQL 将返回第一个没有任何 NULL 列的 UNIQUE 键，作为 PRIMARY KEY
reference_definition	为字段添加注释

以上是创建数据表的一些基础知识，看起来十分复杂，但在实际应用中使用最基本的格式创建数据表即可，具体格式如下：

```
create table table_name (列名 1  属性,列名 2  属性…);
```

例如，使用 CREATE TABLE 语句在 MySQL 数据库 db_admin 中创建一个名为 tb_admin 的数据表，该表包括 id、user、password 和 createtime 等字段，如图 16.13 所示。

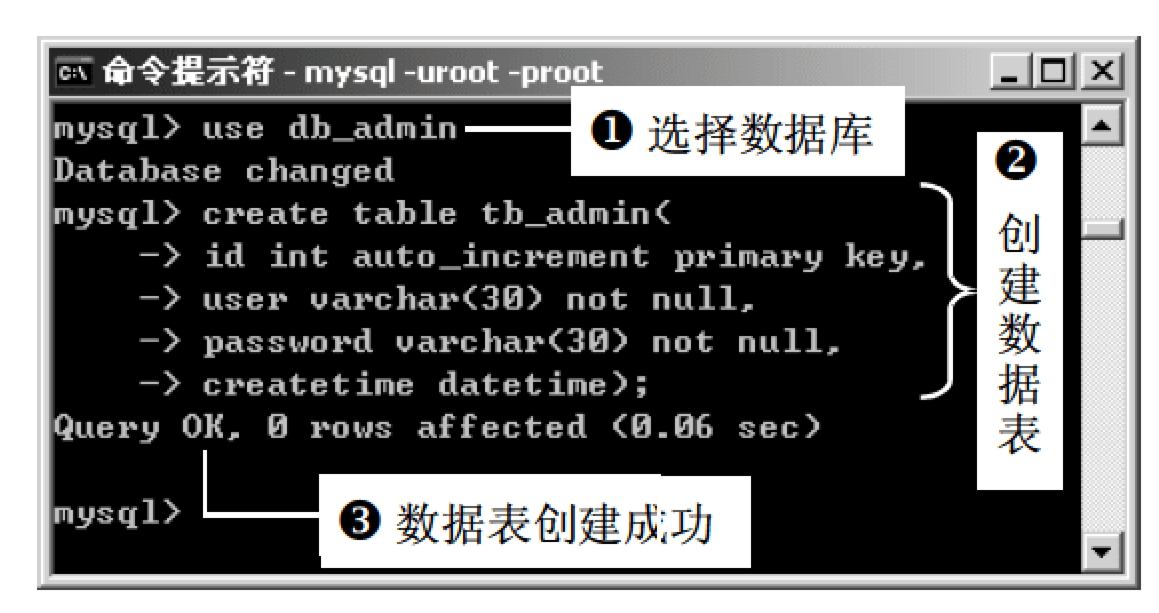


图 16.13　创建 MySQL 数据库

16.4.2　查看表结构 SHOW COLUMNS 或 DESCRIBE

对于创建成功的数据表，可以使用 SHOW COLUMNS 语句或 DESCRIBE 语句查看指定数据表的表结构。下面分别对这两个语句进行介绍。

1．SHOW COLUMNS 语句

SHOW COLUMNS 语句的语法格式如下：

```
SHOW  [FULL] COLUMNS  FROM 数据表名 [FROM 数据库名];
```

或

```
SHOW  [FULL] COLUMNS  FROM 数据表名.数据库名;
```

例如，使用 SHOW COLUMNS 语句查看数据表 tb_admin 表结构，如图 16.14 所示。

2．DESCRIBE 语句

DESCRIBE 语句的语法格式如下：

```
DESCRIBE 数据表名;
```



其中，DESCRIBE 可以简写成 DESC。在查看表结构时，也可以只列出某一列的信息。

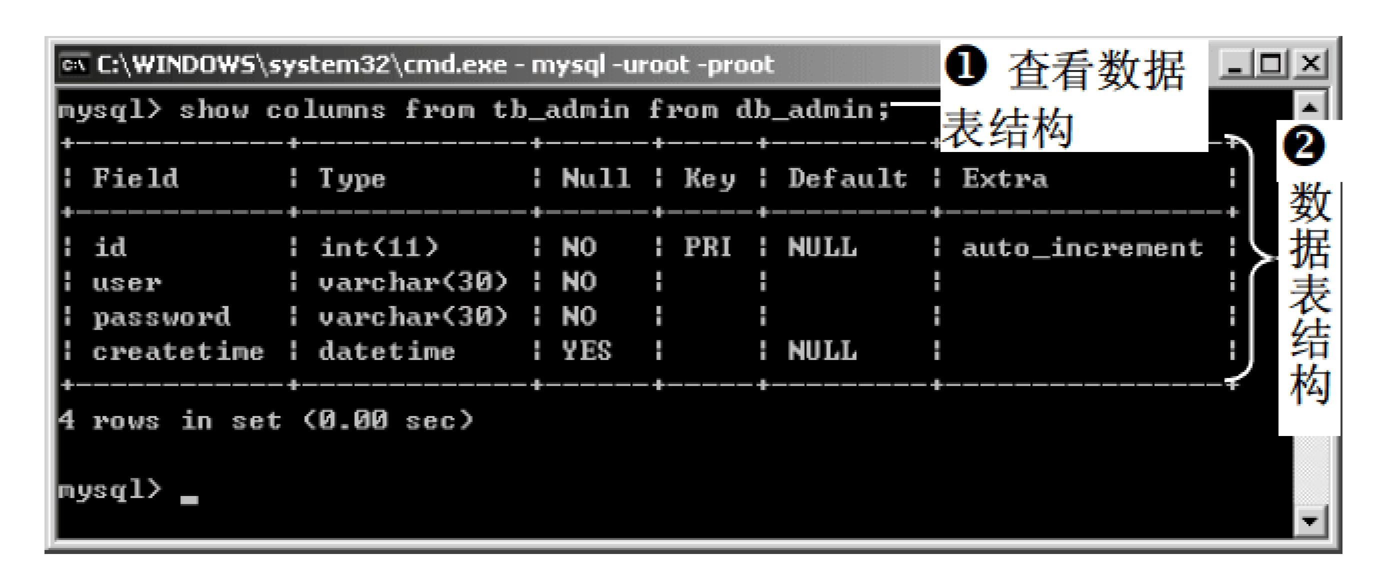


图 16.14　查看表结构

其语法格式如下：

```
DESCRIBE 数据表名 列名;
```

例如，使用 DESCRIBE 语句的简写形式查看数据表 tb_admin 中的某一列信息，如图 16.15 所示。



图 16.15　查看表的某一列信息

16.4.3　修改表结构 ALTER TABLE

修改表结构使用 ALTER TABLE 语句。修改表结构指增加或者删除字段、修改字段名称或者字段类型、设置取消主键外键、设置取消索引以及修改表的注释等。

语法格式如下：

```
ALTER[IGNORE] TABLE 数据表名 alter_spec[,alter_spec]...
```


注意

当指定 IGNORE 时，如果出现重复关键的行，则只执行一行，其他重复的行被删除。

其中，alter_spec 子句定义要修改的内容，其语法如下：

```
alter_specification:
     ADD [COLUMN] create_definition [FIRST | AFTER column_name ]        //添加新字段
  |  ADD INDEX [index_name] (index_col_name,...)                        //添加索引名称
  |  ADD PRIMARY KEY (index_col_name,...)                               //添加主键名称
  |  ADD UNIQUE [index_name] (index_col_name,...)                       //添加唯一索引
```

```
| ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}   //修改字段名称
| CHANGE [COLUMN] old_col_name create_definition                //修改字段类型
| MODIFY [COLUMN] create_definition                             //修改子句定义字段
| DROP [COLUMN] col_name                                        //删除字段名称
| DROP PRIMARY KEY                                              //删除主键名称
| DROP INDEX index_name                                         //删除索引名称
| RENAME [AS] new_tbl_name                                      //更改表名
| table_options
```

ALTER TABLE 语句允许指定多个动作，其动作间使用逗号分隔，每个动作表示对表的一个修改。

例如，添加一个新的字段 email，类型为 varchar(50)，not null，将字段 user 的类型由 varchar(30) 改为 varchar(40)，代码如下：

```
alter table tb_admin add email varchar(50) not null ,modify user varchar(40);
```

在命令行模式下的运行情况如图 16.16 所示。

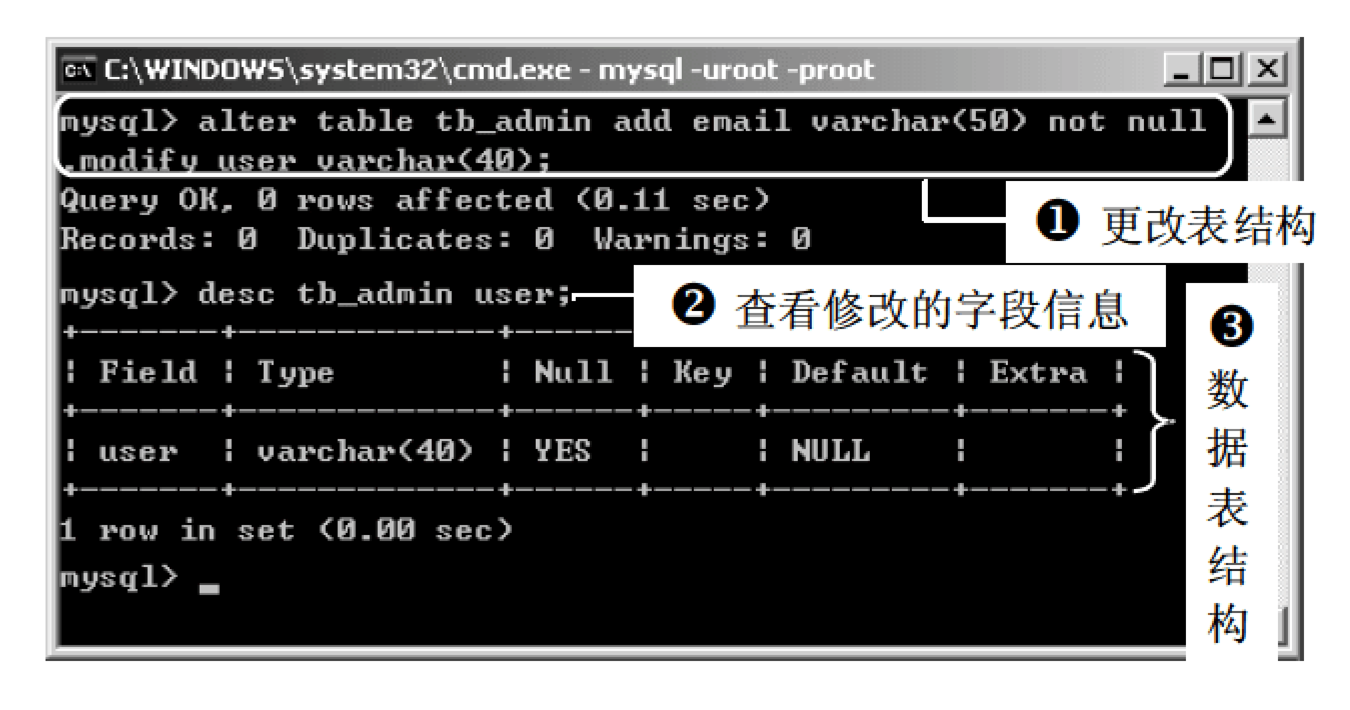


图 16.16　修改表结构

图 16.16 中只给出了修改 user 字段类型的结果，读者可以通过语句 mysql> show tb_admin;查看整个表的结构，以确认 email 字段是否添加成功。



说明

通过 alter 修改表列，其前提是必须将表中数据全部删除，然后才可以修改表列。

16.4.4　重命名表 RENAME TABLE

重命名数据表使用 RENAME TABLE 语句，语法如下：

```
RENAME TABLE 数据表名1 To 数据表名2
```



说明

该语句可以同时对多个数据表进行重命名，多个表之间以逗号“,”分隔。

例如，对数据表 tb_admin 进行重命名，更名后的数据表为 tb_user，如图 16.17 所示。



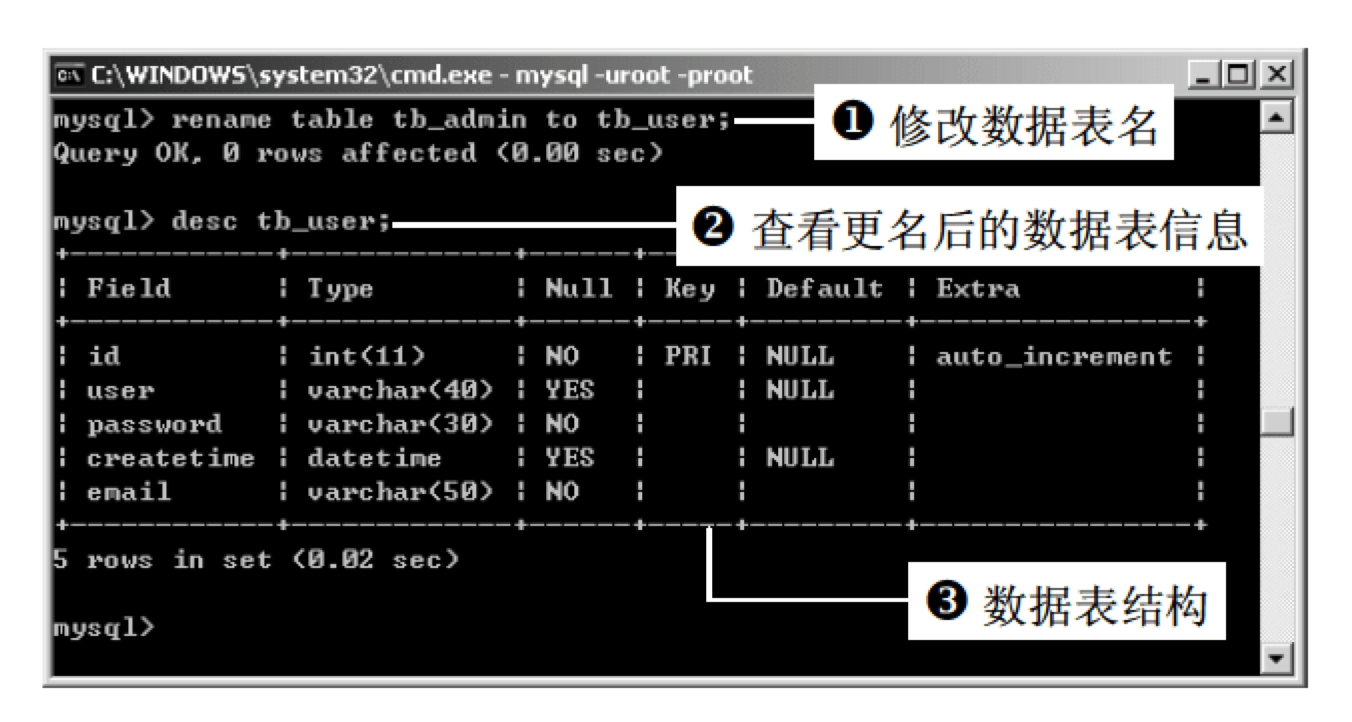


图 16.17　对数据表进行更名

16.4.5　删除表 DROP TABLE

删除数据表的操作很简单，同删除数据库的操作类似，使用 DROP TABLE 语句即可实现。其格式如下：

```
DROP TABLE 数据表名;
```

例如，删除数据表 tb_user，如图 16.18 所示。

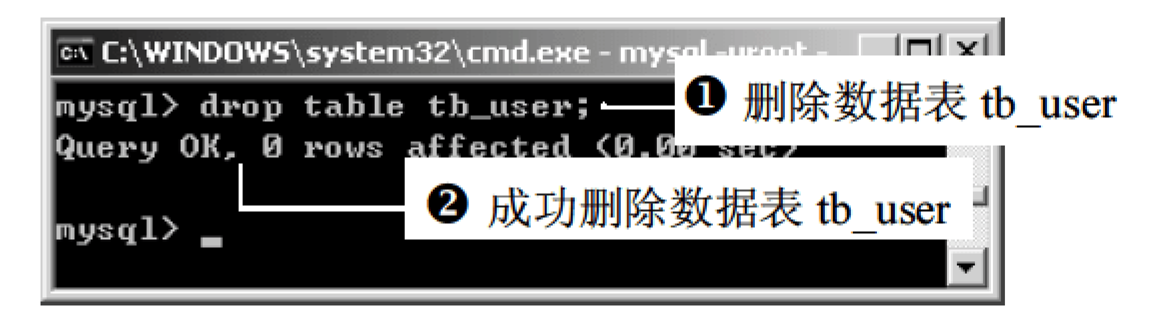


图 16.18　删除数据表

注意

删除数据表的操作应该谨慎使用。一旦删除了数据表，那么表中的数据将会全部清除，没有备份则无法恢复。

在删除数据表的过程中，若删除一个不存在的表将会产生错误，如果在删除语句中加入 IF EXISTS 关键字就不会出错了。格式如下：

```
drop table if exists 数据表名;
```

16.5　MySQL 语句操作

视频讲解：光盘\TM\lx\16\MySQL 语句操作.exe

在数据表中插入、浏览、修改和删除记录可以在 MySQL 命令行中使用 sql 语句完成，下面介绍如

何在 MySQL 命令行中执行基本的 sql 语句。

16.5.1 插入记录 insert

在建立一个空的数据库和数据表时，首先需要考虑的是如何向数据表中添加数据，该操作可以使用 insert 语句来完成。

语法格式如下：

```
insert  into 数据表名(column_name,column_name2, … ) values (value1, value2, … )
```

在 MySQL 中，一次可以同时插入多行记录，各行记录的值清单在 VALUES 关键字后以逗号“,”分隔，而标准的 sql 语句一次只能插入一行。

例如，向管理员信息表 tb_admin 中插入一条数据信息，如图 16.19 所示。



图 16.19 插入记录

16.5.2 查询数据库记录 select

要从数据库中把数据查询出来，就要用到数据查询语句 select。select 语句是最常用的查询语句，它的使用方式有些复杂，但功能强大。

select 语句的语法格式如下：

```
select selection_list                  //要查询的内容，选择哪些列
from 数据表名                           //指定数据表
where primary_constraint               //查询时需要满足的条件，行必须满足的条件
group by grouping_columns              //如何对结果进行分组
order by sorting_cloumns               //如何对结果进行排序
having secondary_constraint            //查询时满足的第二条件
limit count                            //限定输出的查询结果
```

其中使用的子句将在后面逐个介绍。下面先介绍 select 语句的简单应用。

☑ 使用 select 语句查询一个数据表

使用 select 语句时，首先要确定所要查询的列。“*”代表所有列。

例如，查询管理员信息表 tb_admin 中的所有数据，如图 16.20 所示。

这是查询表中所有列的操作，还可以针对表中的某一列或多列进行查询。

☑ 查询表中的一列或多列

针对表中的多列进行查询，只要在 select 后面指定要查询的列名即可，多列之间用“,”分隔。



例如，查询管理员信息表 tb_admin 中的 id（ID 编号）、user（用户名）、password（用户密码）和 email（用户邮箱）字段，并指定查询条件为用户 ID 编号为 1，如图 16.21 所示。



图 16.20 查询数据表的全部数据



图 16.21 查看数据表中指定字段的数据

☑ 多表查询

针对多个数据表进行查询，关键是 where 子句中查询条件的设置，要查找的字段名最好用“表名.字段名”表示，这样可以防止因表之间字段重名而造成无法获知该字段属于哪个表，在 where 子句中多个表之间所形成的联动关系应按如下形式书写：

```
表 1.字段=表 2.字段 and 其他查询条件
```

多表查询的 sql 语句格式如下：

```
select 字段名 from 表 1,表 2…… where 表 1.字段=表 2.字段 and 其他查询条件
```

例如，查询学生表和成绩表，查询条件是学生表的 userid 等于成绩表的 sid，并且学生的 userid 等于 001。其代码如下：

```
select * from tb_student,tb_sscore where tb_student.userid=tb_sscore.sid and tb_student.userid=001
```

说明

有关 select 语句，其应用的形式很多，这里介绍的只是其中最简单的内容，感兴趣的读者可以对其进行深入的研究。对于 sql 语句的使用，是一个很丰富的知识点，合理地运用 sql 语句，能够提高程序查询、输出数据的速度。由于本书以 PHP 语言的学习为主，所以这里没有对 sql 语句的知识进行深入的讲解。

16.5.3　修改记录 update

要执行修改的操作可以使用 update 语句，该语句的格式如下：

```
update 数据表名 set column_name = new_value1,column_name2 = new_value2, …where condition
```

其中，set 子句指出要修改的列和它们给定的值，where 子句是可选的，如果给出它将指定记录中哪行应该被更新，否则，所有的记录行都将被更新。

例如，下面将管理员信息表 tb_admin 中用户名为 tsoft 的管理员密码 111 修改为 896552，如图 16.22 所示。

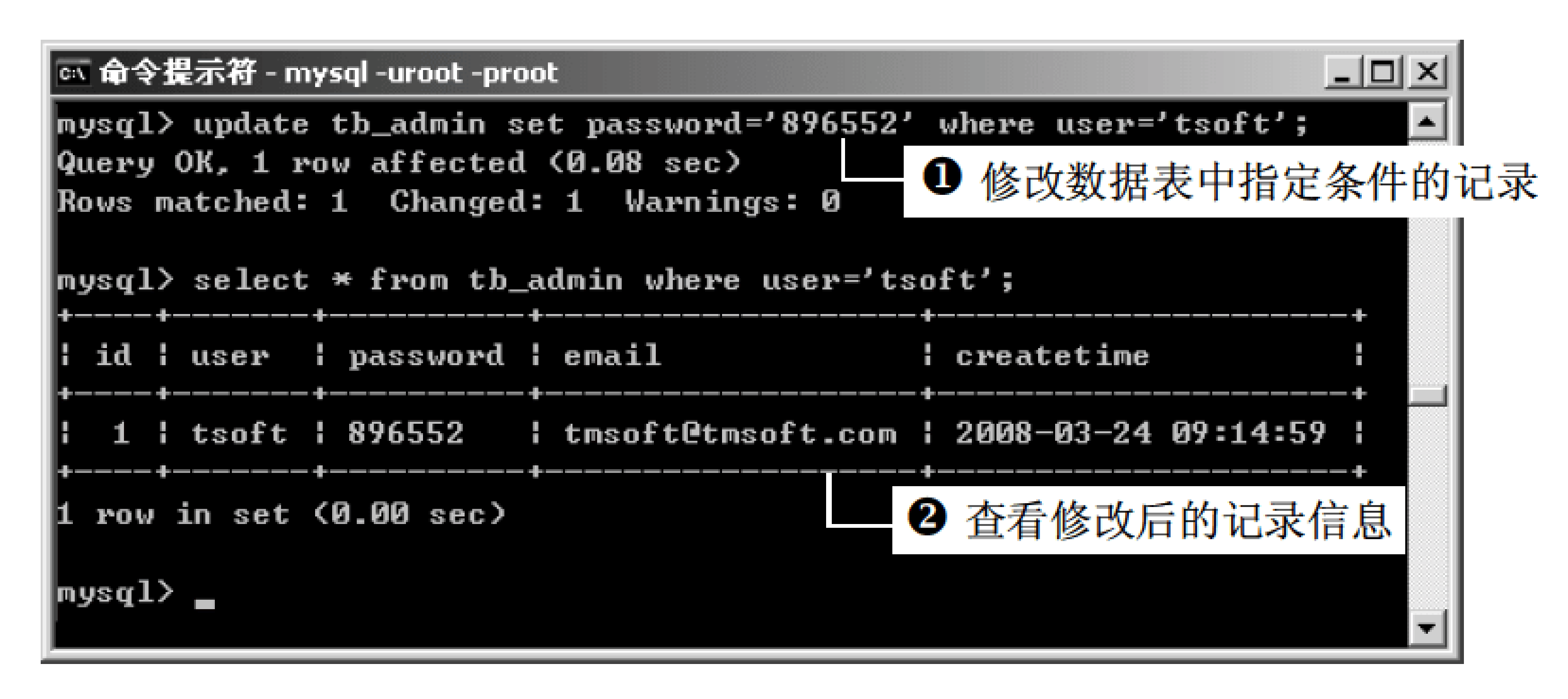


图 16.22　修改指定条件的记录


注意

更新时一定要保证 where 子句的正确性，一旦 where 子句出错，将会破坏所有改变的数据。

16.5.4　删除记录 delete

在数据库中，有些数据已经失去意义或者错误时就需要将它们删除，此时可以使用 delete 语句，该语句的格式如下：

```
delete from 数据表名 where condition
```


注意

该语句在执行过程中，如果没有指定 where 条件，将删除所有的记录；如果指定了 where 条件，将按照指定的条件进行删除。

例如，删除管理员数据表 tb_admin 中用户名为“小欣”的记录信息，如图 16.23 所示。





图 16.23　删除数据表中指定的记录

注意

在实际的应用中，执行删除操作时，执行删除的条件一般应该为数据的 id，而不是具体某个字段值，这样可以避免一些不必要的错误发生。

16.6　MySQL 数据库备份和恢复

在前面的章节中，已经对 MySQL 数据库、数据表的各种操作进行了详细的讲解，下面介绍如何实现对 MySQL 数据库中的数据进行备份和恢复。

16.6.1　数据的备份

在命令行模式下完成对数据的备份，使用的是 MYSQLDUMP 命令。通过该命令可以将数据以文本文件的形式存储到指定的文件夹下。

说明

要在命令行模式下操作 MySQL 数据库，必须要对电脑的环境变量进行设置，右击“我的电脑”，在弹出的快捷菜单中选择“属性”命令，在弹出的对话框中选择“高级”选项卡，然后在新弹出的对话框中单击“环境变量”按钮，在用户变量的文本框中找到变量 PATH 并选中，单击“编辑”按钮，在变量 PATH 的变量值文本框中添加“D:\webpage\AppServ\MySQL\bin”（MySQL 数据库中 bin 文件夹的安装路径），然后单击“确定”按钮。其中添加的 bin 文件夹的路径根据自己安装 MySQL 数据库的位置而定。

注意

如果使用集成化的安装包来配置 PHP 的开发环境，那么就不需要进行上述的配置操作，因为集成化安装包已经自行配置完成。但是，如果是独立安装的 MySQL，那么就必须进行上述的配置，才能在命令行模式下操作 MySQL 数据库。

通过 MYSQLDUMP 命令备份整个数据库的操作步骤如下：

（1）选择“开始”/“运行”命令，如图 16.24 所示。

（2）在如图 16.25 所示的对话框中输入“cmd”，单击“确定”按钮，进入命令行模式。




图 16.24　选择“运行”命令



图 16.25　进入命令行模式

（3）在命令行模式中直接输入“mysqldump –uroot –proot db_database16 >F:\db_database16.txt”，然后按 Enter 键即可，如图 16.26 所示。

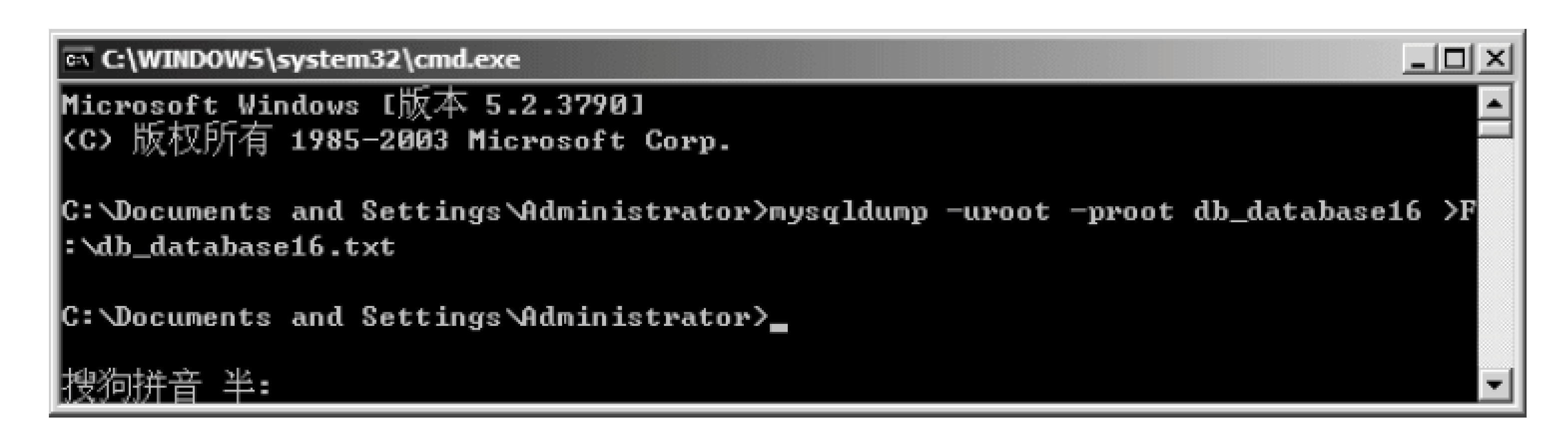


图 16.26　通过命令备份 db_database16 数据库

其中，-uroot 中的“root”是用户名，而-proot 中的“root”是密码，“db_database16”是数据库名，“F:\db_database16.txt”是数据库备份存储的位置。

最后，可以查看一下，在这个文件夹是否存在备份的数据库文件。



在输入命令的过程中，“-uroot”中是没有空格的，在该命令的结尾处也没有任何的结束符，只要按 Enter 键即可。

16.6.2　数据的恢复

既然可以对数据库进行备份，那么就一定可以对数据库文件进行恢复操作。执行数据库的恢复操作使用的是 MySQL 命令。其命令格式如下：

```
mysql -uroot –proot db_database <F:\db_database16.txt”
```

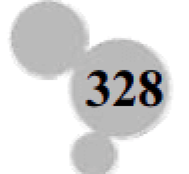
其中 mysql 是使用的命令，–u 后的 root 代表用户名，–p 后的 root 代表密码，db_database 代表数据库名（或表名），“<”号后面的“F:\db_database16.txt”是数据库备份文件存储的位置。

下面介绍实现数据库恢复的操作步骤：

（1）选择“开始”/“运行”命令。

（2）在弹出的对话框中输入“cmd”，单击“确定”按钮，进入命令行模式。

（3）在命令行模式中直接输入“mysql –uroot –proot db_database16 <F:\db_database16.txt”，然后



按 Enter 键即可，如图 16.27 所示。

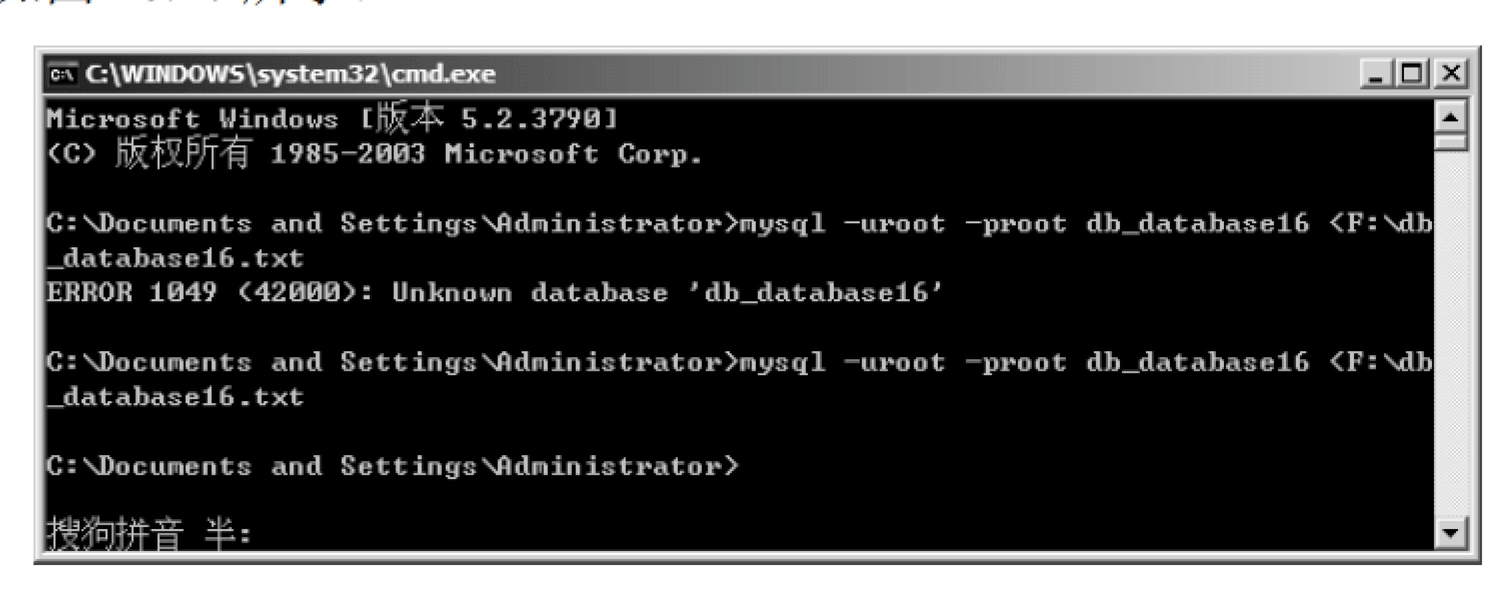


图 16.27　通过命令恢复 db_database16 数据库

其中，-uroot 中的“root”是用户名，而-proot 中的“root”是密码，“db_database16”是要恢复的数据库名，“F:\db_database16.txt”是数据库备份文件存储的位置。

说明

在进行数据库的恢复时，在 MySQL 数据库中必须存在一个空的、将要恢复的数据库，否则就会出现图 16.27 中第一次执行恢复操作时的错误。

最后，可以查看一下，数据库是否恢复成功，如图 16.28 所示。

```
C:\WINDOWS\system32\cmd.exe - mysql -uroot -proot
Microsoft Windows [版本 5.2.3790]
(C) 版权所有 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>mysql -uroot -proot db_database16 <F:\db
_database16.txt
ERROR 1049 (42000): Unknown database 'db_database16'

C:\Documents and Settings\Administrator>mysql -uroot -proot db_database16 <F:\db
_database16.txt

C:\Documents and Settings\Administrator>mysql -uroot -proot
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 38
Server version: 5.0.51b-community-nt-log MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use db_database16
Database changed
mysql> select * from tb_user;
+----+--------+--------+
| id | user   | pass   |
+----+--------+--------+
|  1 | mrsoft | 123456 |
|  2 | tm     | 111    |
+----+--------+--------+
2 rows in set (0.00 sec)

+----+--------+--------+
```

图 16.28　查看数据库

16.7　小　　结

本章主要介绍 MySQL 数据库的基本操作，包括创建、查看、选择、删除数据库；创建、修改、更名、删除数据表；插入、浏览、修改、删除记录以及数据库的备份和恢复，这些是程序开发人员必

须掌握的内容。如果用户不习惯在命令提示符下管理数据库，第 17 章将要介绍 MySQL 数据库的图形化管理工具 phpMyAdmin，则能够使读者在可视化的图形工具中轻松操作和管理数据库。另外，本章还介绍了启动、连接、断开和停止 MySQL 服务器的方法，要求读者熟练掌握。

16.8 练习与实践

1．创建一个数据库 db_shop，然后查看 MySQL 服务器中所有的数据库，确认数据库 db_shop 是否创建成功。如果该数据库成功创建，则选择该数据库并进行删除操作。**（答案位置：光盘\TM\sl\16\1）**

2．在数据库 db_shop 中，按如图 16.29 所示的表结构创建商品信息表 tb_shangpin。**（答案位置：光盘\TM\sl\16\2）**

Field	Type	Null	Key	Default	Extra
id	int(4)	NO	PRI	NULL	auto_increment
mingcheng	varchar(25)	YES		NULL	
jianjie	mediumtext	YES		NULL	
addtime	varchar(25)	YES		NULL	
dengji	varchar(5)	YES		NULL	
xinghao	varchar(25)	YES		NULL	
tupian	varchar(200)	YES		NULL	
shuliang	int(4)	YES		NULL	
typeid	int(4)	YES		NULL	
huiyuanjia	varchar(25)	YES		NULL	
pinpai	varchar(25)	YES		NULL	

图 16.29　创建表结构

3．将会员信息表 tb_shangpin 更名为 tb_shop，并确认更名操作是否成功。**（答案位置：光盘\TM\sl\16\3）**

4．向商品信息表 tb_shop 的各字段中添加 10 条商品信息。**（答案位置：光盘\TM\sl\16\4）**

5．浏览商品信息表 tb_shop 中的全部数据，将第一条数据的商品名称修改为“数码相机”，将该表中的最后一条数据删除。**（答案位置：光盘\TM\sl\16\5）**

第17章

phpMyAdmin图形化管理工具

（视频讲解：56分钟）

安装MySQL数据库后，用户即可在命令提示符下进行创建数据库和数据表等各种操作，这种方法非常麻烦，而且需要有专业的SQL语言知识。目前，官方应用PHP开发了一个类似于SQL Server的可视化图形管理工具phpMyAdmin。该工具可以运行在各种版本的PHP及MySQL下。通过phpMyAdmin完全可以对数据库进行各种操作，如建立、复制和删除数据等。phpMyAdmin为初学者提供了图形化的操作界面，这样MySQL数据库的创建就不必在命令提示符下通过命令实现，从而大大提高了程序开发的效率。

通过阅读本章，您可以：

- 熟悉创建、修改和删除数据库的方法
- 掌握创建、修改和删除数据表的方法
- 灵活运用SQL语句操作数据表
- 熟练使用phpMyAdmin向数据表中插入数据
- 熟练使用phpMyAdmin浏览数据表中的数据
- 熟练使用phpMyAdmin搜索数据表中的数据
- 掌握使用phpMyAdmin生成和执行MySQL数据库脚本的方法

17.1 phpMyAdmin 介绍

phpMyAdmin 是众多 MySQL 图形化管理工具中使用最广泛的一种，是一款使用 PHP 开发的 B/S 模式的 MySQL 客户端软件，该工具是基于 Web 跨平台的管理程序，并且支持简体中文。用户可以在官方网站 www.phpmyadmin.net 上免费下载到最新的版本。phpMyAdmin 为 Web 开发人员提供了类似于 Access、SQL Server 的图形化数据库操作界面，通过该管理工具可以对 MySQL 进行各种操作，如创建数据库、数据表和生成 MySQL 数据库脚本文件等。

注意

如果使用集成化安装包来配置 PHP 的开发环境，就无须单独下载 phpMyAdmin 图形化管理工具，因为集成化的安装包中大多包括图形化管理工具。

17.2 phpMyAdmin 的使用

视频讲解：光盘\TM\lx\17\phpMyAdmin 的使用.exe

无论是 Windows 操作系统还是 Linux 操作系统，phpMyAdmin 图形化管理工具的使用方法都是一样的。下面讲解在 phpMyAdmin 图形化管理工具的可视化界面中操作数据库及数据表。

17.2.1 操作数据库

在浏览器地址栏中输入 http://localhost/phpMyAdmin/，进入 phpMyAdmin 主界面，接下来即可进行 MySQL 数据库的操作，下面将分别介绍如何创建、修改和删除数据库。

1. 创建数据库

在 phpMyAdmin 的主界面中，首先选择 Language 下拉列表框中的“中文-Chinese simplified（简体中文）”选项，然后在下拉列表框中选择所要使用的编码，一般选择 gb2312_chinese_ci 简体中文编码格式，在文本框中输入数据库的名称“db_study”，再选择数据库使用的编码类型 gb2312_chinese_ci，单击“创建”按钮，创建数据库，如图 17.1 所示。成功创建数据库后，将显示如图 17.2 所示的界面。

说明

在右侧界面中可以对该数据库进行各种操作，如结构、SQL、导出、搜索、查询和删除等，单击相应的超链接进入相应的操作界面。但在只创建了数据库还没有创建数据表的情况下，只能够执行结构、SQL、Import、操作、权限和删除 6 项操作，其他 3 项操作 搜索、查询、导出 不能执行，当指向其超链接时，显示不可用标记。

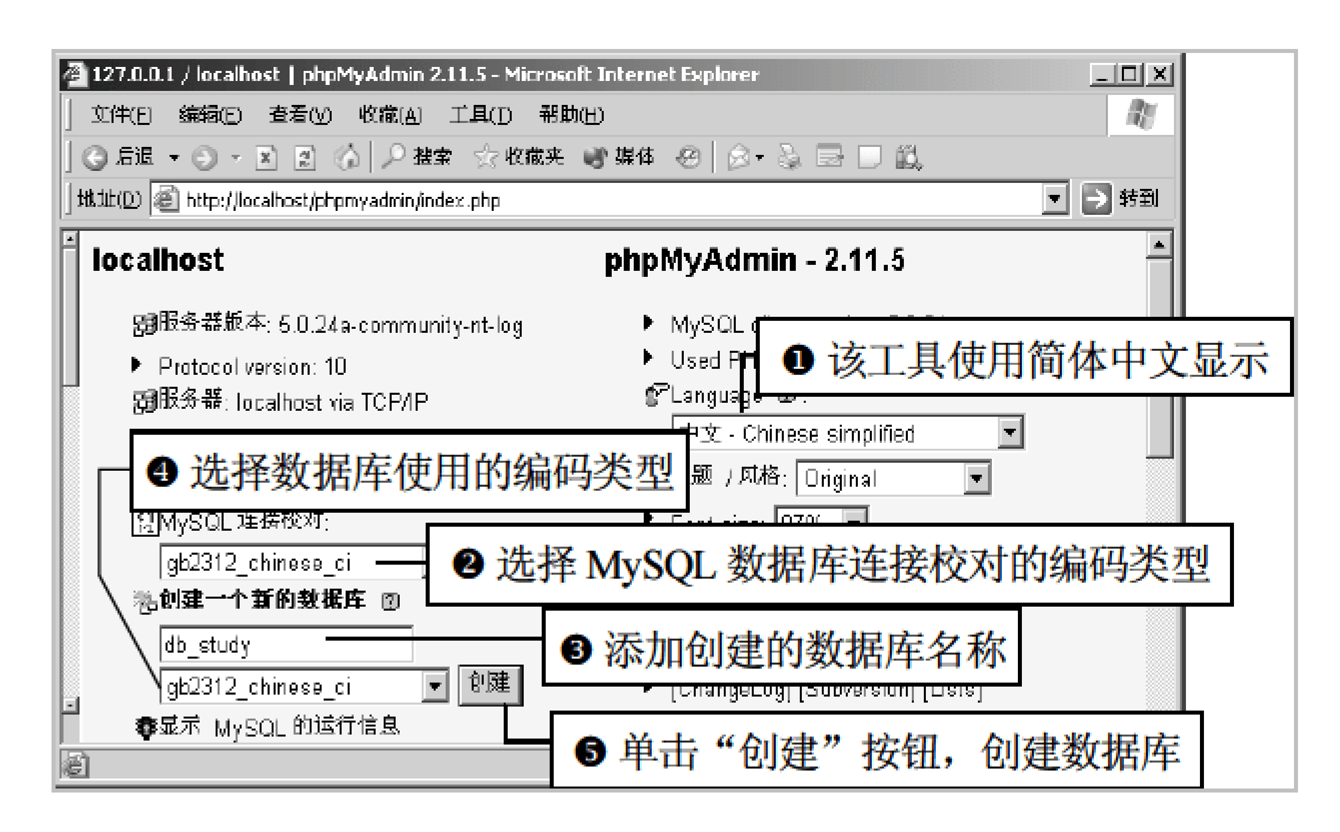


图 17.1　phpMyAdmin 管理主界面

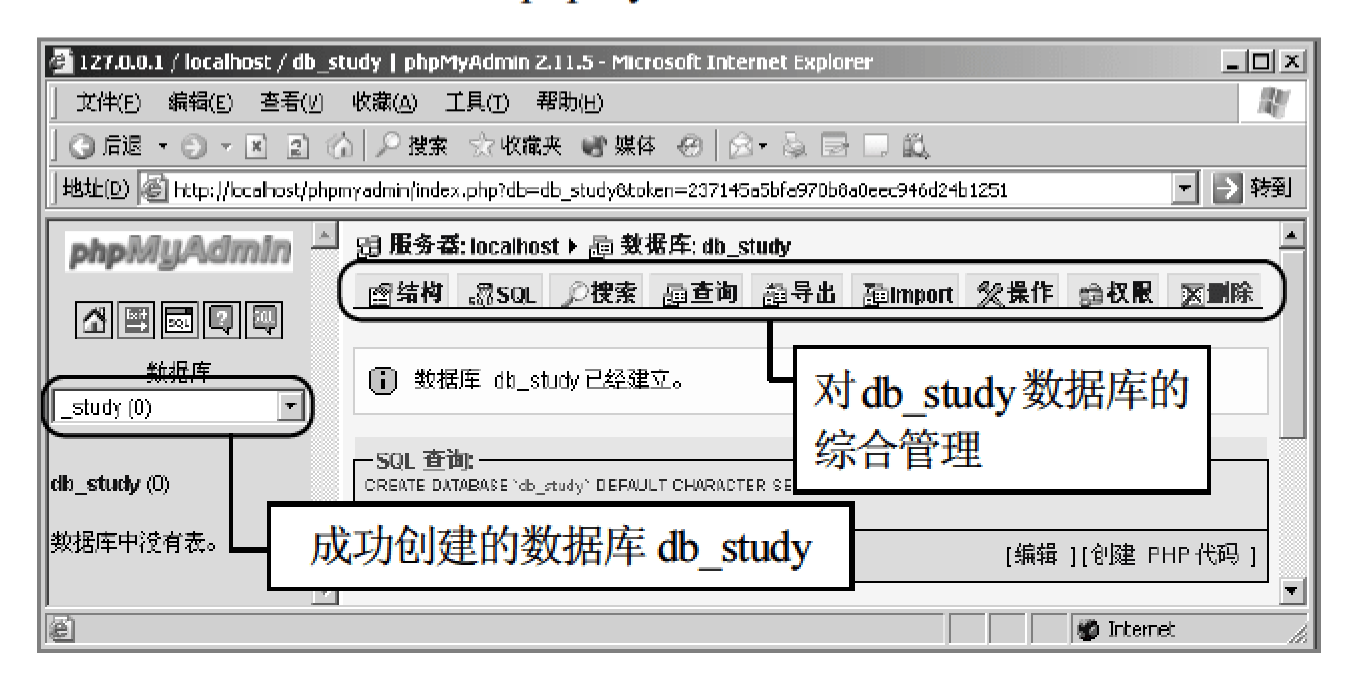


图 17.2　成功创建数据库

2. 修改数据库

在如图 17.2 所示的界面右侧，可以对当前数据库进行修改。单击界面中的“操作”超链接，进入修改操作页面。

☑ 可以对当前数据库执行创建数据表的操作。在创建数据表提示信息下的两个文本框中分别输入要创建的数据表的名称和字段总数，单击“执行”按钮，进入创建数据表结构页面，具体创建方法将在 17.2.2 节中进行详细讲解。

☑ 可以对当前的数据库重命名，在“重新命名数据库为”文本框中输入新的数据库名称，单击“执行”按钮，即可成功修改数据库名称。

修改数据库的效果如图 17.3 所示。

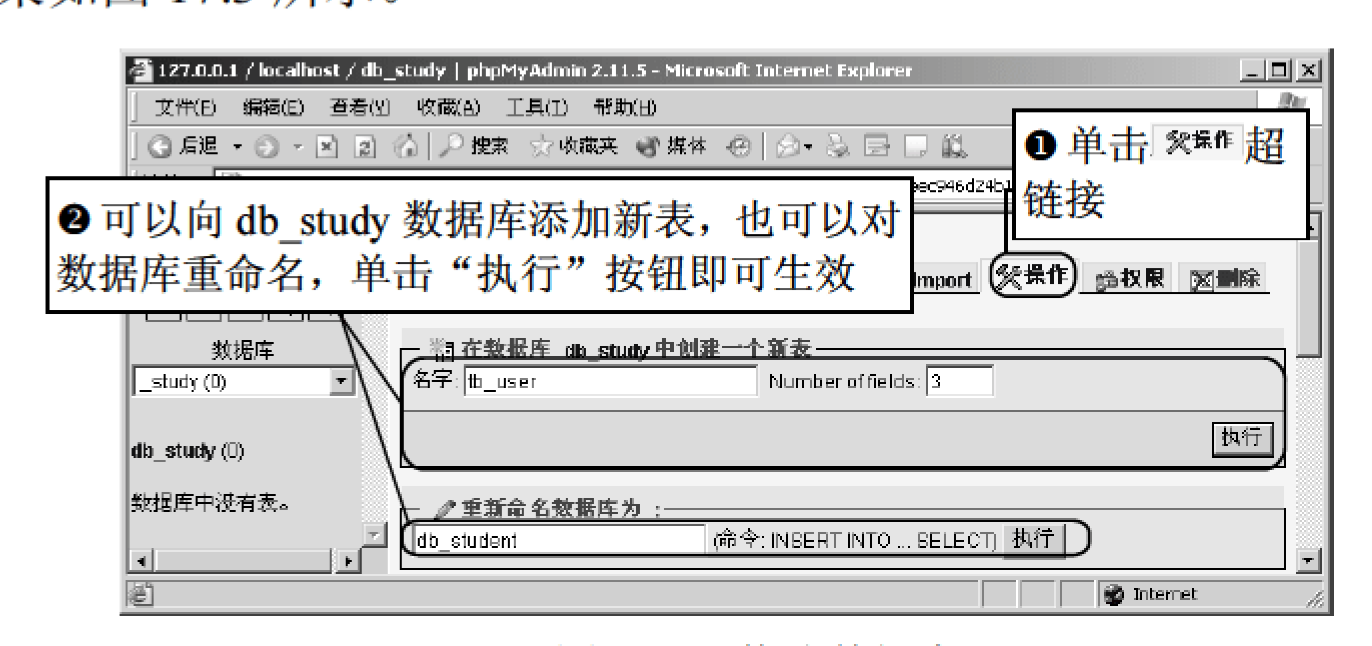


图 17.3　修改数据库

3. 删除数据库

要删除某个数据库，首先在左侧的下拉列表框中选择该数据库，然后单击右侧界面中的删除超链接（如图 17.3 所示）即可成功删除指定的数据库。

17.2.2 操作数据表

操作数据表是以选择指定的数据库为前提，然后在该数据库中创建并管理数据表。下面介绍如何创建、修改和删除数据表。

1. 创建数据表

下面以管理员信息表 tb_admin 为例，讲解数据表的创建方法。

创建数据库 db_study 后，在右侧的操作页面中输入数据表的名称和字段数，然后单击“执行”按钮，即可创建数据表，如图 17.4 所示。

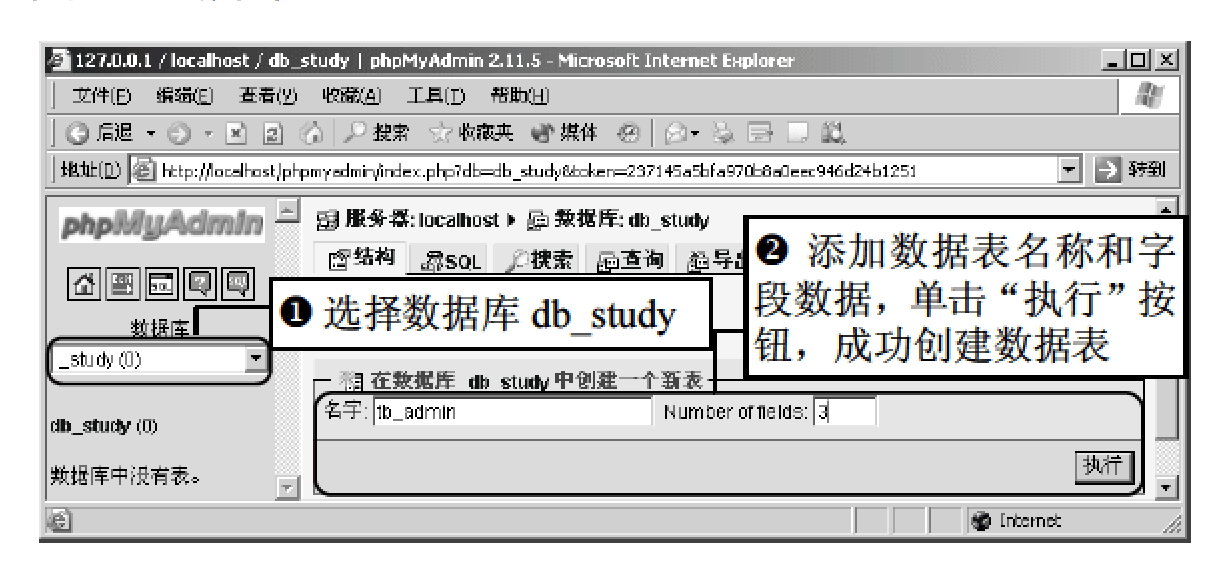


图 17.4　创建数据表

成功创建数据表 tb_admin 后，将显示数据表结构界面。在表单中输入各个字段的详细信息，包括字段名、数据类型、长度/值、编码格式、是否为空和主键等，以完成对表结构的详细设置。当所有的信息都输入完成以后，单击“保存”按钮，创建数据表结构，如图 17.5 所示。成功创建数据表结构后，将显示如图 17.6 所示的界面。

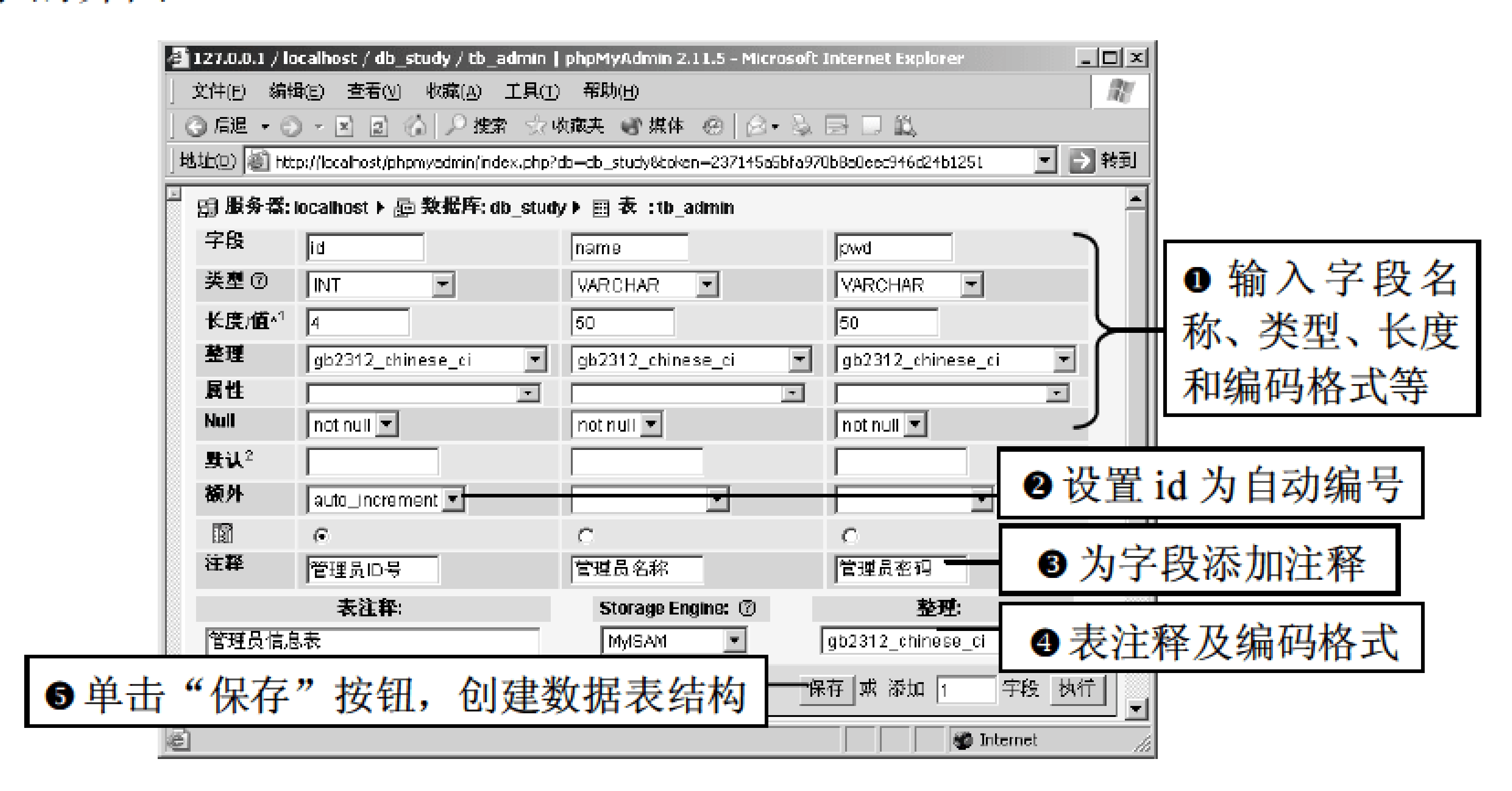


图 17.5　创建数据表结构





单击“执行”按钮，可以将数据表结构以横版显示，便于编辑表结构。



图 17.6　成功创建数据表

2．修改数据表

一个新的数据表被创建后，进入到数据表页面中，在这里可以通过改变表的结构来修改表，可以执行添加列、删除列、索引列、修改列的数据类型或者字段的长度/值等操作，如图 17.7 所示。



图 17.7　修改数据表结构

3．删除数据表

要删除某个数据表，首先在左侧的下拉列表框中选择该数据库，在指定的数据库中选择要删除的数据表，然后单击右侧界面中的删除超链接（如图 17.7 所示），即可成功删除指定的数据表。

17.2.3　使用 SQL 语句操作数据表

单击 phpMyAdmin 主界面中的SQL超链接，打开 SQL 语句编辑区，输入完整的 SQL 语句，来实现数据的查询、添加、修改和删除操作。

1．使用 SQL 语句插入数据

在 SQL 语句编辑区中使用 insert 语句向数据表 tb_admin 中插入数据，单击“执行”按钮，向数据表中插入一条数据，如图 17.8 所示。如果提交的 SQL 语句有错误，系统会给出警告，提示用户修改；如果提交的 SQL 语句正确，则弹出如图 17.9 所示的提示信息。

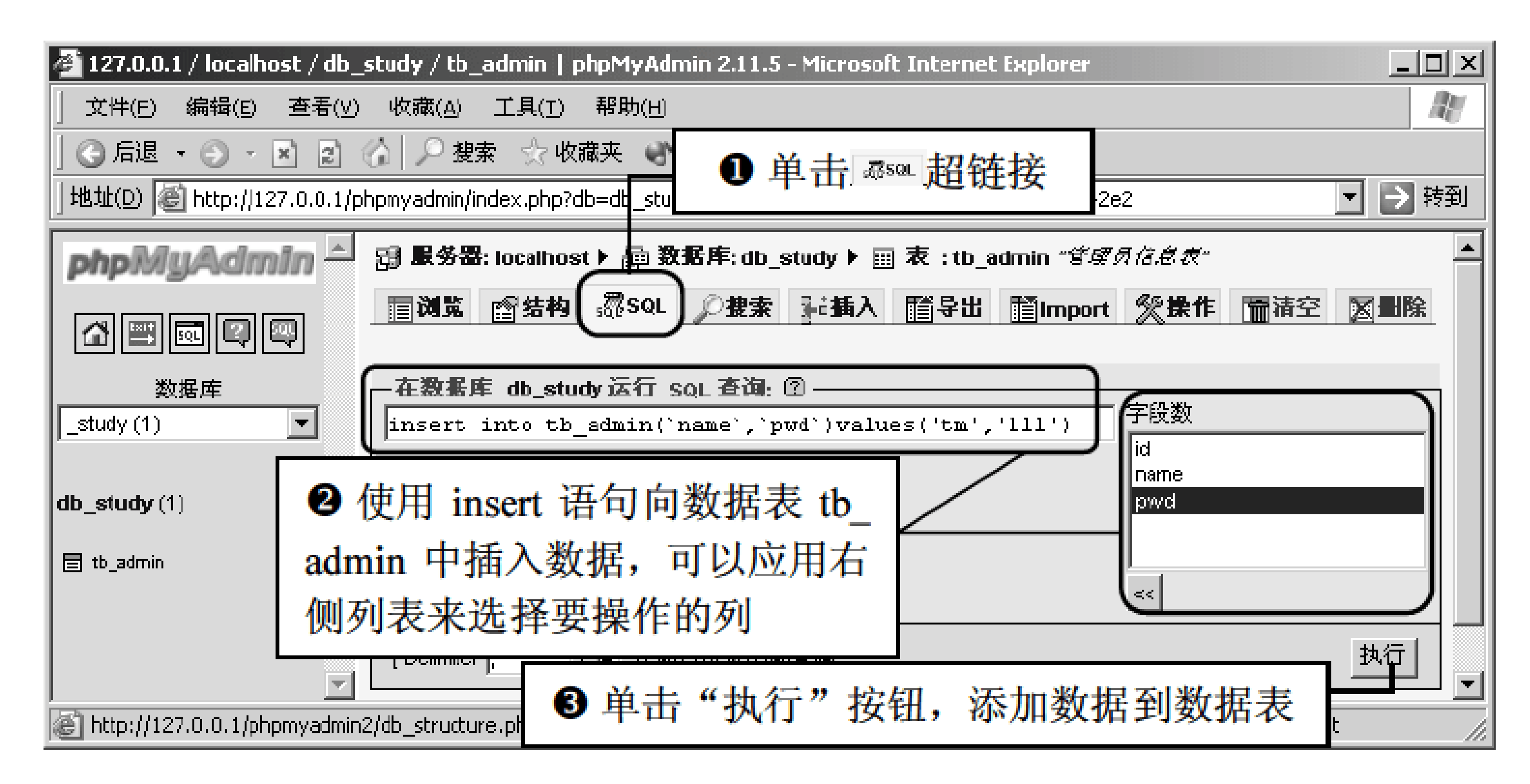


图 17.8　使用 SQL 语句向数据表中插入数据



←T→	id	name	pwd
	1	tm	111

图 17.9　成功添加数据信息



为了编写方便，可以利用其右侧的属性列表来选择要操作的列，只要选中要添加的列，双击其选项或者单击“<<”按钮添加列名称。

2. 使用 SQL 语句修改数据

在 SQL 语句编辑区使用 update 语句修改数据信息，将 id 为 1 的管理员的名称改为“纯净水”，密码改为“111”，添加的 SQL 语句如图 17.10 所示。

单击“执行”按钮，该语句的实现过程如图 17.11 所示。

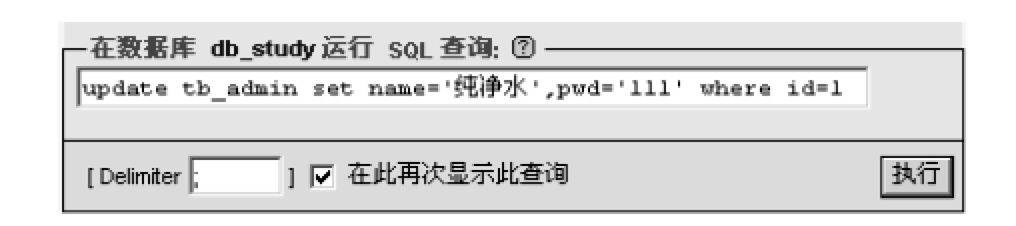


图 17.10　添加修改数据信息的 SQL 语句

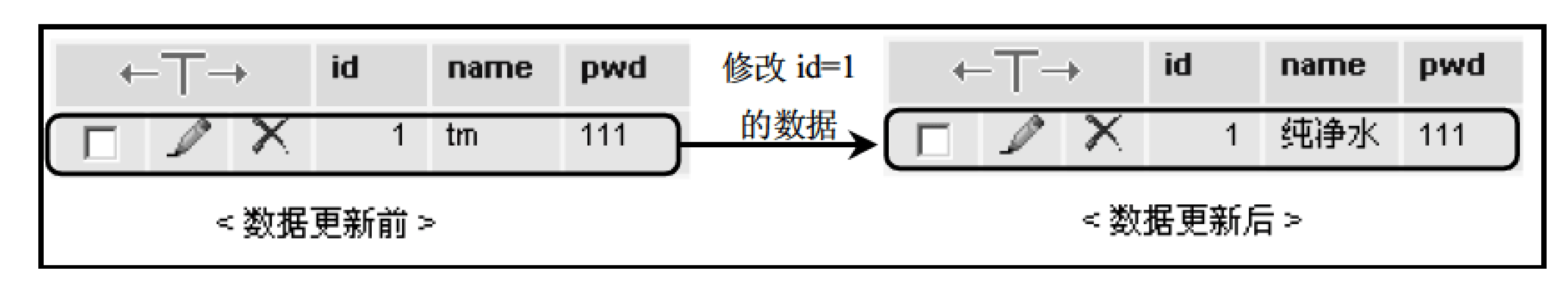
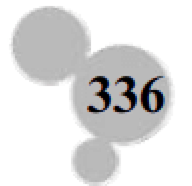


图 17.11　修改单条数据的实现过程

3. 使用 SQL 语句查询数据

在 SQL 语句编辑区使用 select 语句检索指定条件的数据信息，将 id 小于 4 的管理员全部显示出来，添加的 SQL 语句如图 17.12 所示。



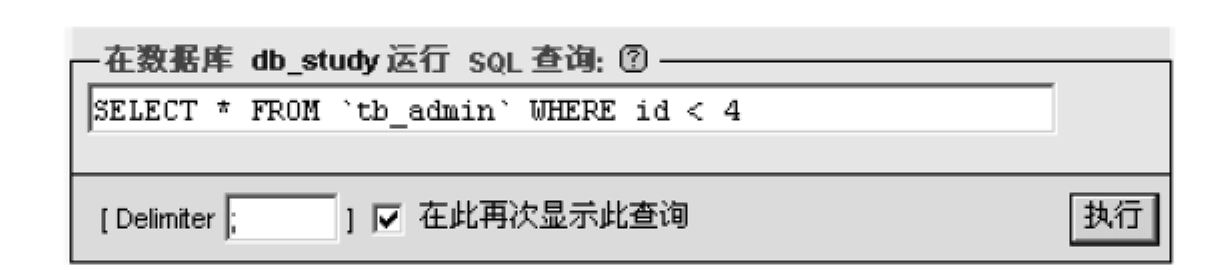


图 17.12　添加查询数据信息的 SQL 语句

单击“执行”按钮，该语句的实现过程如图 17.13 所示。

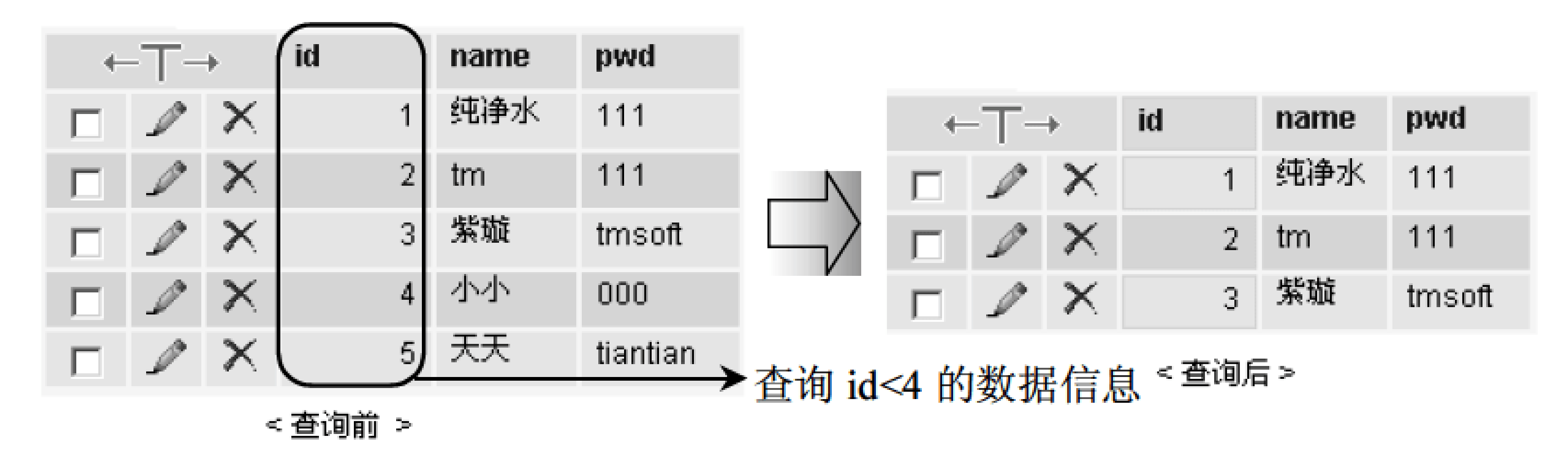


图 17.13　查询指定条件的数据信息的实现过程

除了对整个表的简单查询外，还可以进行一些复杂的条件查询(使用 where 子句提交 LIKE、ORDER BY、GROUP BY 等条件查询语句）及多表查询。

4．使用 SQL 语句删除数据

在 SQL 语句编辑区使用 delete 语句检索指定条件的数据或全部数据信息，删除名称为 tm 的管理员信息，添加的 SQL 语句如图 17.14 所示。

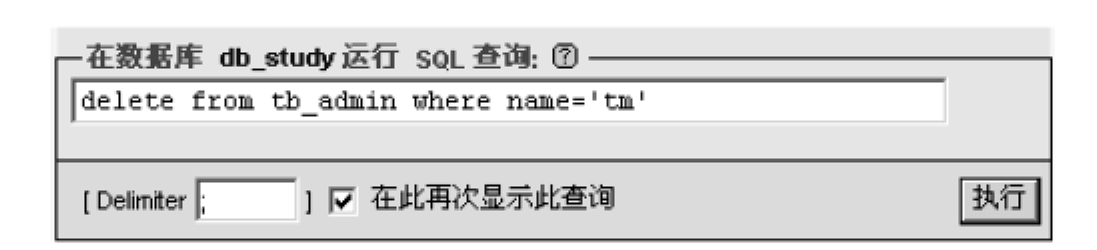


图 17.14　添加删除指定数据信息的 SQL 语句

注意

如果 delete 语句后面没有 where 条件值，那么将删除指定数据表中的全部数据。

单击“执行”按钮，弹出确认删除操作对话框，单击“确定”按钮，执行数据表中指定条件的删除操作。该语句的实现过程如图 17.15 所示。

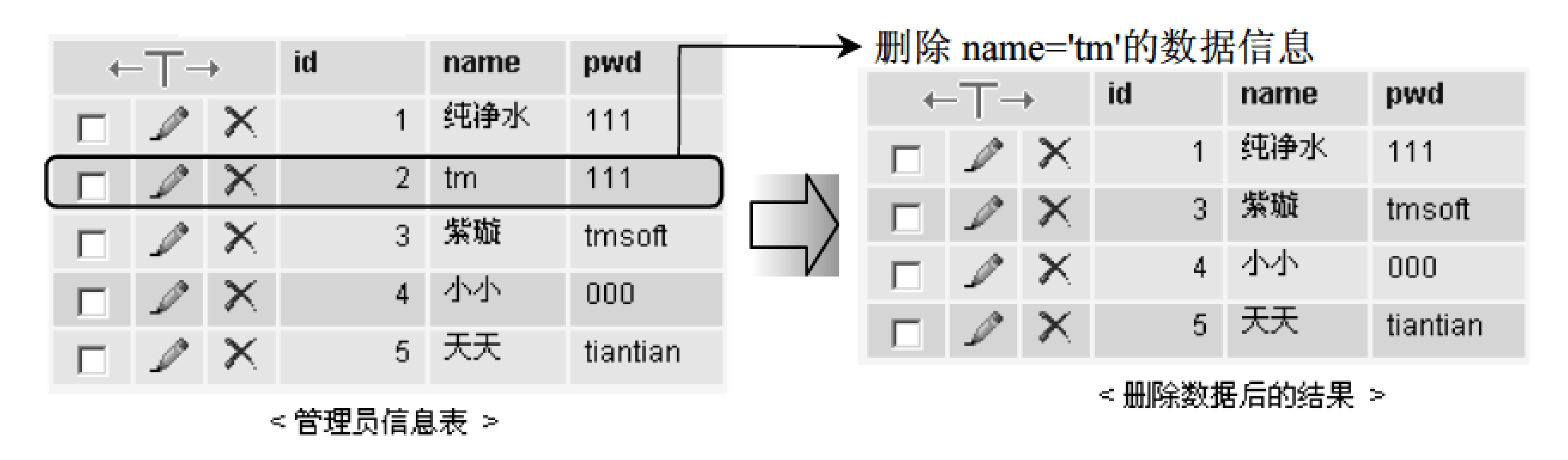


图 17.15　删除指定条件的数据信息的实现过程

17.2.4　管理数据记录

在创建完数据库和数据表后，可以通过操作数据表来管理数据。下面分别介绍插入数据、浏览数据、搜索数据的方法。

1．插入数据

选择某个数据表后，单击插入超链接，进入插入数据界面，如图 17.16 所示。在界面中输入各字段值，单击“执行”按钮即可插入记录。在默认情况下，一次可以插入两条记录。

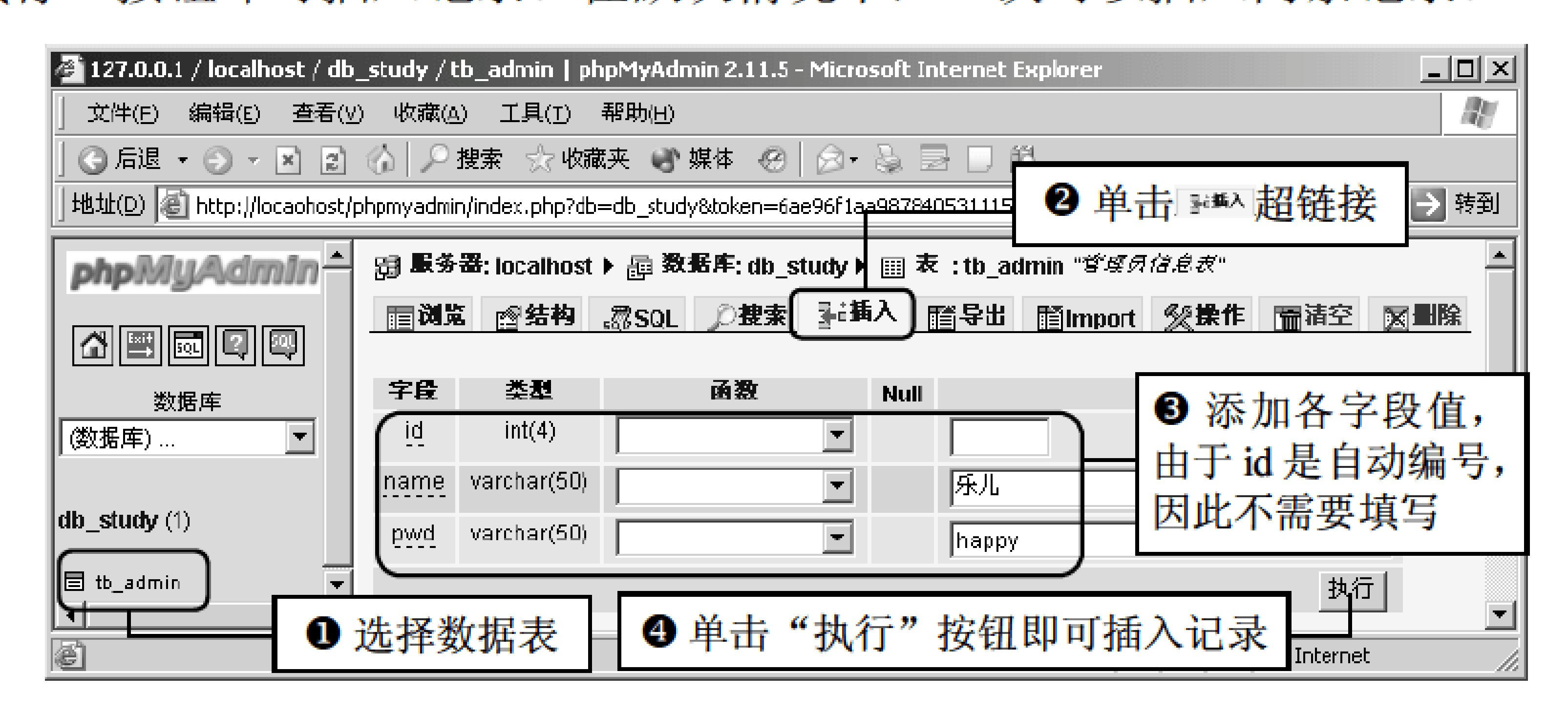


图 17.16　插入数据

2．浏览数据

选择某个数据表后，单击浏览超链接进入浏览界面，如图 17.17 所示。单击每行记录中的✎按钮，可以对该记录进行编辑；单击每行记录中的✕按钮，可以删除该条记录。

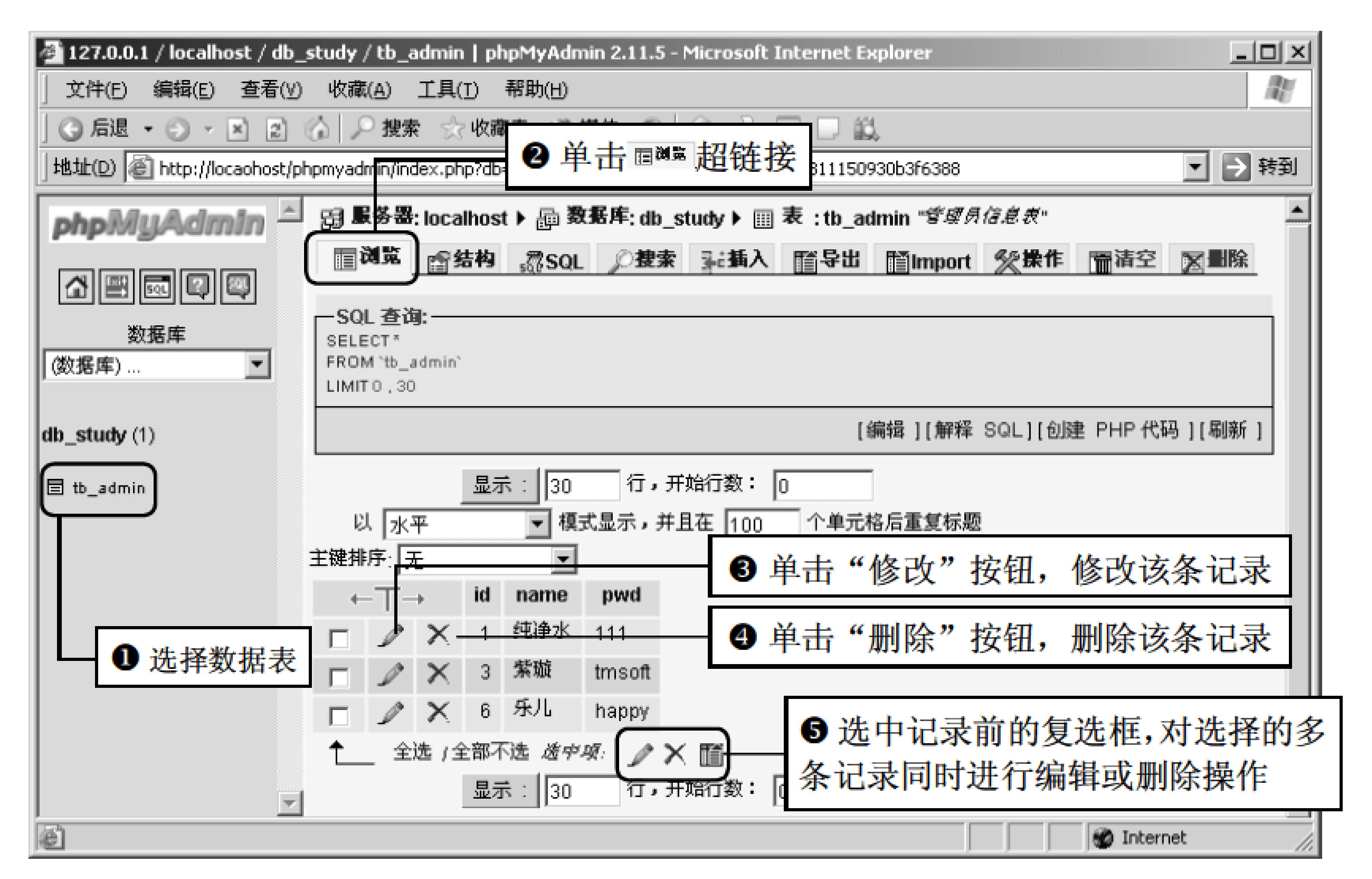
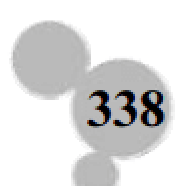


图 17.17　浏览数据



3．搜索数据

选择某个数据表后，单击搜索超链接进入搜索页面，如图 17.18 所示。在该页面中，可以在选择字段的列表框中选择一个或多个列。如果要选择多个列，先按下 Ctrl 键，然后依次单击要选择的字段名，查询结果将按照选择的字段名进行输出。

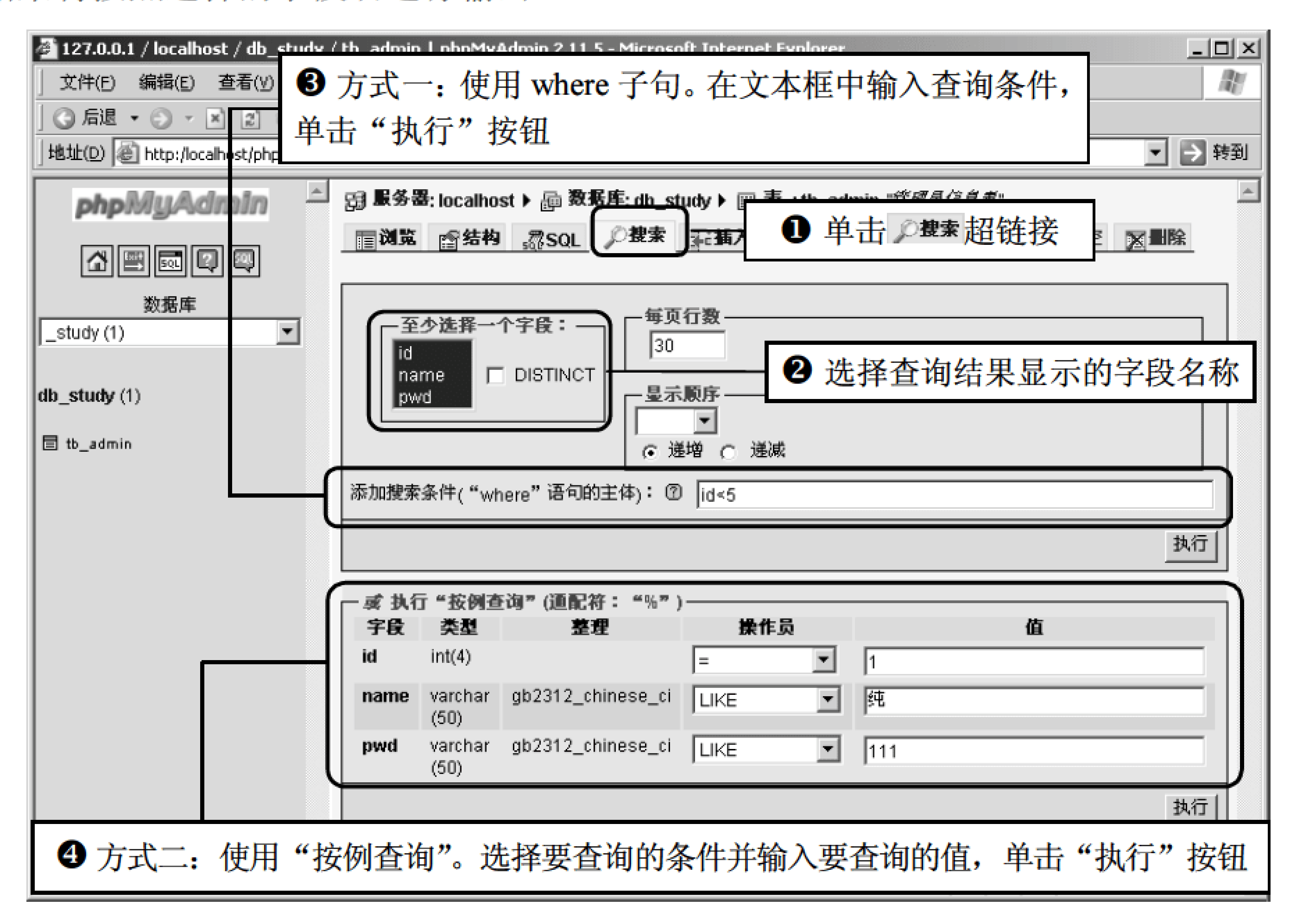


图 17.18　搜索查询

在该界面中可以对记录按条件进行查询。查询方式有两种：第一种方式选择构建 where 语句查询，直接在“添加搜索条件”文本框中输入查询语句，然后单击其后的“执行”按钮。第二种方式使用按例查询，选择查询的条件，并在文本框中输入要查询的值，单击“执行”按钮。

17.2.5　生成和执行 mysql 数据库脚本

生成和执行 mysql 数据库脚本是互逆的两个操作，执行 mysql 脚本是通过生成的扩展名为.sql 文件导入数据记录到数据库中；生成 mysql 脚本是将数据表结构、表记录存储为.sql 的脚本文件。可以通过生成和执行 mysql 脚本实现数据库的备份和还原操作。下面分别介绍生成和执行 mysql 数据库脚本的方法。

1．生成 mysql 数据库脚本

单击 phpMyAdmin 主界面中的导出超链接，打开导出编辑区，如图 17.19 所示。选择导出文件的格式，这里使用默认选项 SQL，选中“另存为文件”复选框，单击“执行”按钮，弹出如图 17.20 所示的文件下载对话框，单击“保存”按钮，将脚本文件以.sql 格式存储在指定位置。

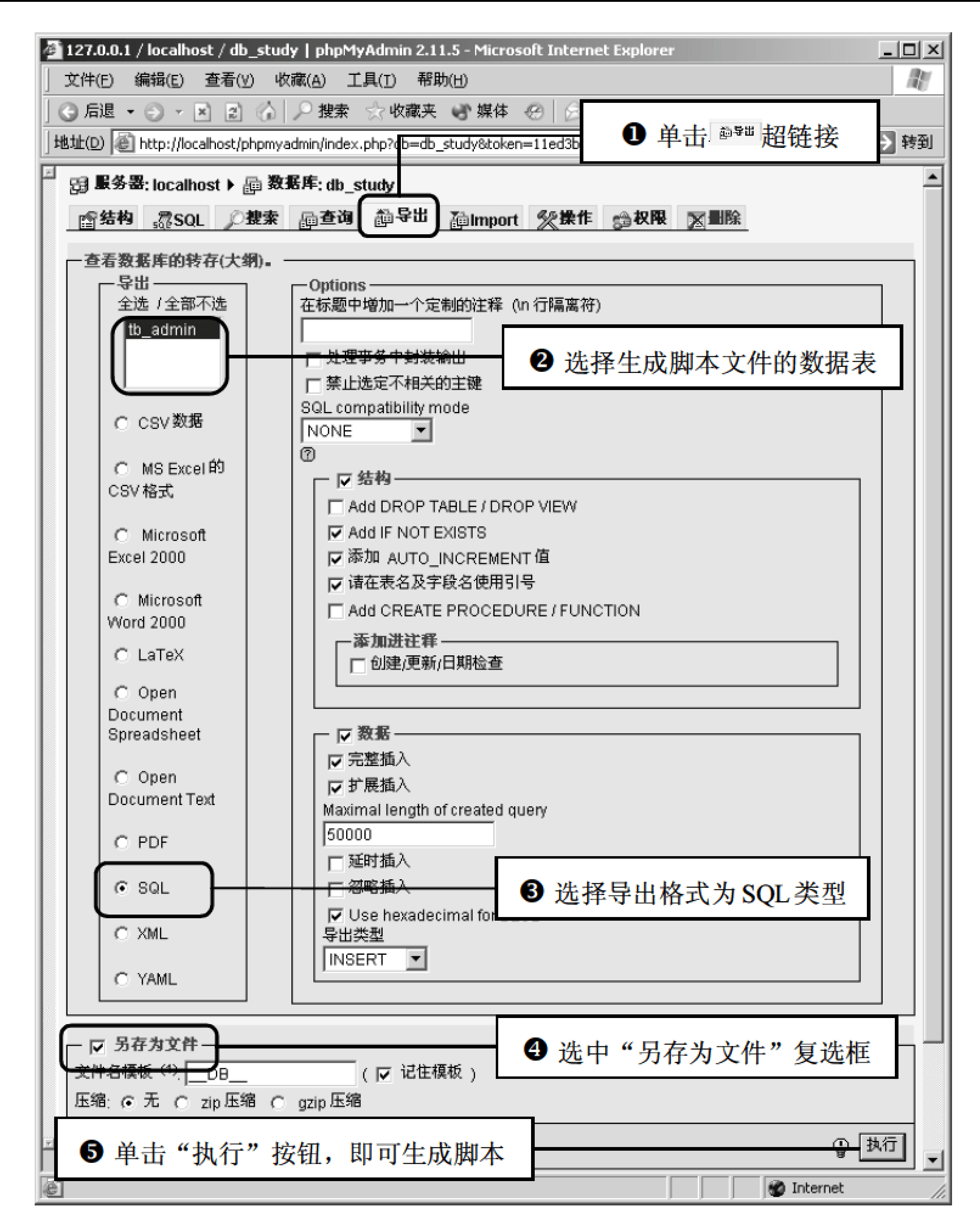


图 17.19　生成 mysql 脚本文件设置界面

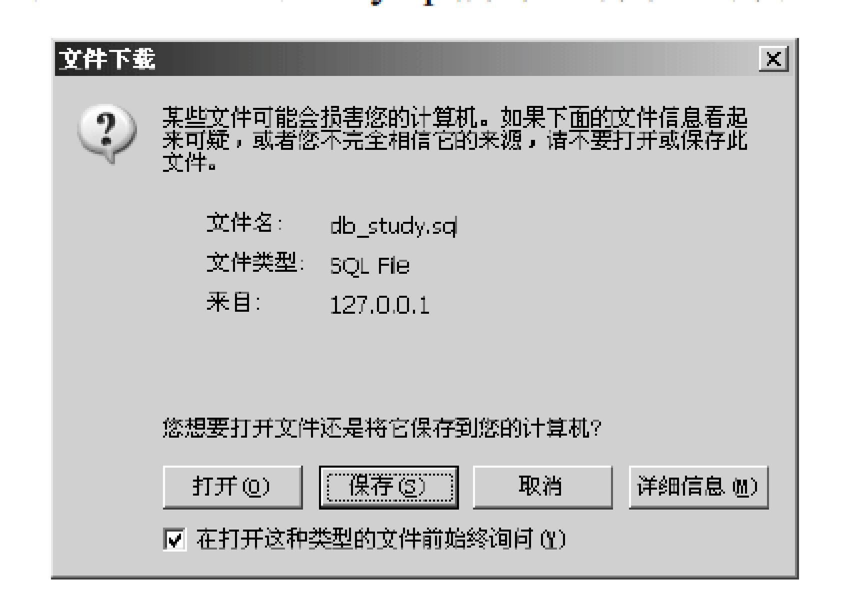
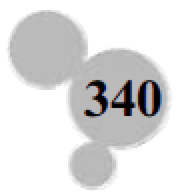


图 17.20　存储 mysql 脚本对话框

2. 执行 mysql 数据库脚本

单击 Import 超链接，进入执行 mysql 数据库脚本界面，单击“浏览”按钮查找脚本文件（如 db_study.sql）所在位置，如图 17.21 所示，单击“执行”按钮，即可执行 mysql 数据库脚本文件。



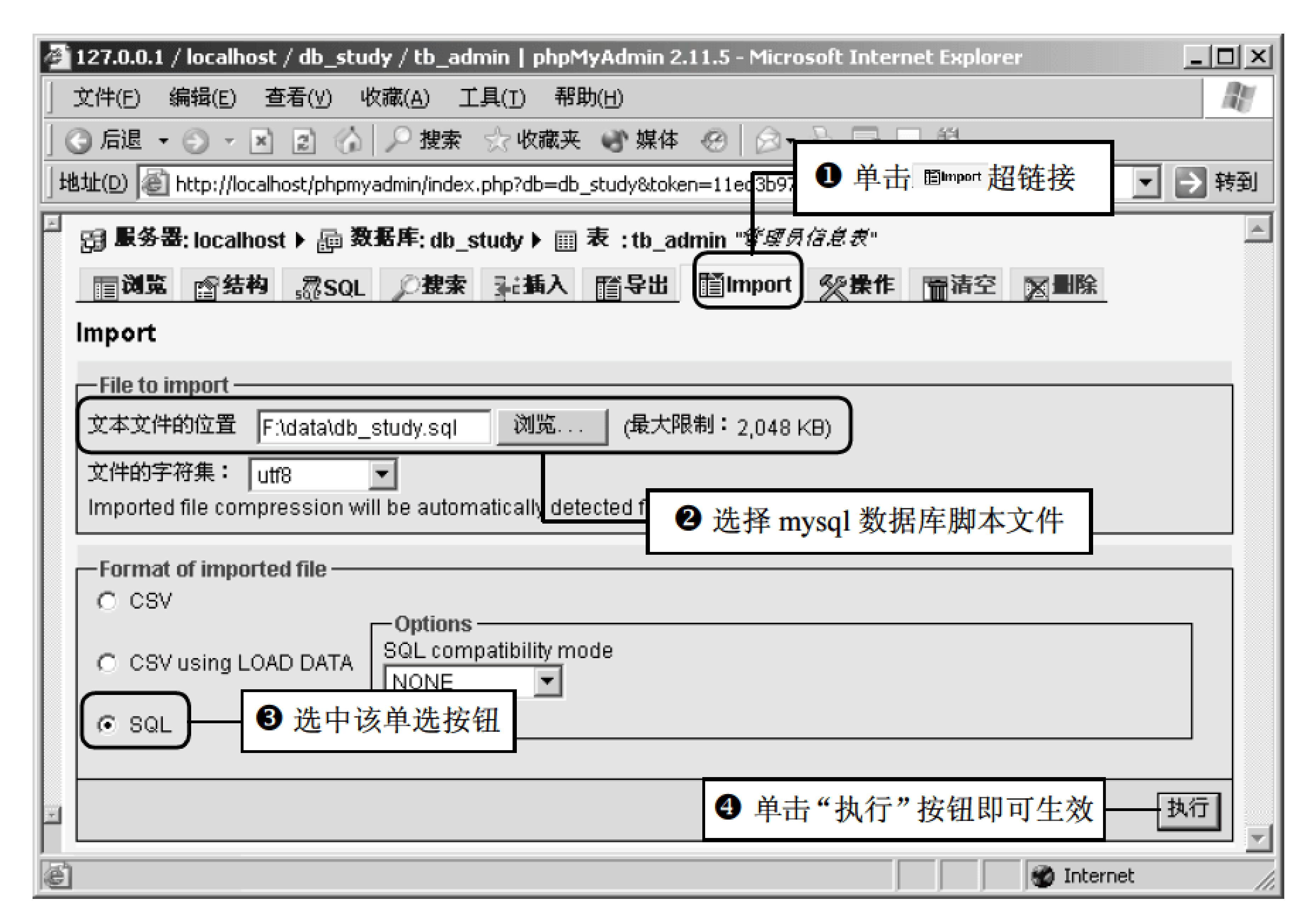


图 17.21　执行 mysql 数据库脚本文件

注意

在执行 mysql 脚本文件前，首先应确保数据库中存在与所导入数据库同名的数据库，如果没有同名的数据库，则首先要在数据库中创建一个名称与数据文件中的数据库名相同的数据库，然后再执行 MySQL 数据库脚本文件。另外，在当前数据库中，不能有与将要导入数据库中的数据表重名的数据表存在，如果有重名的表存在，导入文件就会失败，并提示错误信息。

说明

单击 phpMyAdmin 图形化工具左侧的SQL按钮，在打开的对话框中单击“导入文件”超链接，然后选择脚本文件所在的位置，也可执行脚本文件。

17.3　小　　结

phpMyAdmin 是提供 MySQL 数据库管理和操作的可视化工具，可以方便地对 MySQL 数据库进行管理。通过本章的学习，读者可以独立安装和配置 phpMyAdmin V2.11.5 版本，并且可以摆脱在命令提示符下创建数据库和数据表的瓶颈，使用可视化的工具 phpMyAdmin 轻松地管理数据库和数据表。对于大型的网站，可生成和执行 MySQL 数据库脚本来维护网站数据库。

17.4　练习与实践

1．创建一个数据库 db_shop，并修改数据库的名称为 shop。**（答案位置：光盘\TM\sl\17\1）**

2．在数据库 shop 中添加两个数据表，在数据表中尝试添加各种数据类型的字段，设置每个表中的 id 为自动编号，并设置为主键。**（答案位置：光盘\TM\sl\17\2）**

3．使用 SQL 语句向数据表中添加字段值。**（答案位置：光盘\TM\sl\17\3）**

4．将数据库生成 SQL 脚本文件 data.sql，然后建立一个数据库 db_library，将生成的脚本文件导入到该数据表中。**（答案位置：光盘\TM\sl\17\4）**

第18章

PHP 操作 MySQL 数据库

（视频讲解：1 小时 6 分钟）

任何一种编程语言都需要对数据库进行操作，PHP 语言也不例外，现在的数据库有很多，如 Oracle，Sybase，SQL Server，MySQL 等。在各种数据库中，MySQL 由于其免费、跨平台、使用方便、访问效率较高等优点而获得了广泛应用。很多中型网站都使用 PHP+MySQL 这一最佳搭档。本章主要讲解如何运用 PHP 操作 MySQL 数据库。

通过阅读本章，您可以：

- 了解 PHP 访问 MySQL 数据库的一般步骤
- 掌握 PHP 连接 MySQL 数据库的方法
- 掌握选择 MySQL 数据库的方法
- 掌握 PHP 执行 SQL 语句的方法
- 应用多种方法获取结果集
- 获取查询结果集中的记录行数
- 解决截取中文字符串乱码的问题
- 掌握面向过程分页显示数据信息的方法
- 通过面向对象的方法操作 MySQL 数据库

18.1　PHP 访问 MySQL 数据库的一般步骤

视频讲解：光盘\TM\lx\18\PHP 访问 MySQL 数据库的一般步骤.exe

MySQL 是一款广受欢迎的数据库，由于它是开源的半商业软件，所以市场占有率高，备受 PHP 开发者的青睐，一直被认为是 PHP 的最佳搭档。PHP 具有强大的数据库支持能力，本节主要讲解 PHP 访问 MySQL 数据库的基本思路。

PHP 访问 MySQL 数据库的一般步骤如图 18.1 所示。

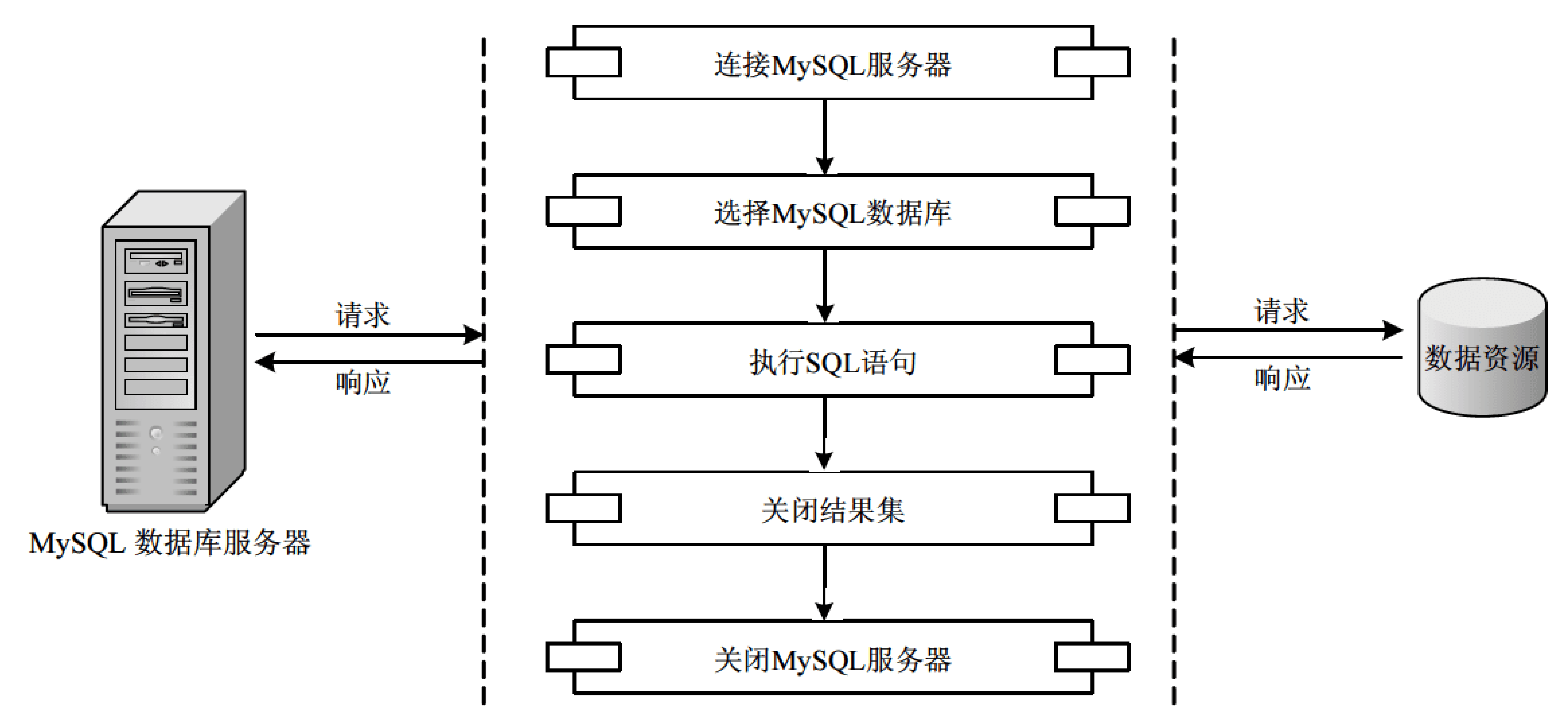


图 18.1　PHP 访问 MySQL 数据库的一般步骤

1. 连接 MySQL 服务器

使用 mysql_connect()函数建立与 MySQL 服务器的连接。有关 mysql_connect()函数的使用可参考本章 18.2.1 节相关内容。

2. 选择 MySQL 数据库

使用 mysql_select_db()函数选择 MySQL 数据库服务器上的数据库，并与数据库建立连接。有关 mysql_select_db()函数的使用可参考本章 18.2.2 节相关内容。

3. 执行 SQL 语句

在选择的数据库中使用 mysql_query()函数执行 SQL 语句。对数据的操作方式主要包括 5 种方式，下面分别介绍。

☑　查询数据：使用 select 语句实现数据的查询功能。

☑　显示数据：使用 select 语句显示数据的查询结果。

☑　插入数据：使用 insert into 语句向数据库中插入数据。



☑　更新数据：使用 update 语句修改数据库中的记录。
☑　删除数据：使用 delete 语句删除数据库中的记录。

有关 mysql_query()函数的使用可参考本章 18.2.3～18.2.7 节相关内容。

4．关闭结果集

数据库操作完成后，需要关闭结果集，以释放系统资源，语法如下：

```
mysql_free_result($result);
```



技巧

如果在多个网页中都要频繁进行数据库访问，则可以建立与数据库服务器的持续连接来提高效率。因为每次与数据库服务器的连接需要较长的时间和较大的资源开销，持续的连接相对来说会更有效。建立持续连接的方法就是在数据库连接时，调用函数 mysql_pconnect()代替 mysql_connect()函数。建立的持续连接在本程序结束时，不需要调用 mysql_close()来关闭。下次程序在此执行 mysql_pconnect()函数时，系统自动直接返回已经建立的持续连接的 ID 号，而不再去真的连接数据库。

有关 mysql_free_result()函数的使用可参考本章 18.2.3～18.2.7 节相关内容。

5．关闭 MySQL 服务器

每使用一次 mysql_connect()或 mysql_query()函数，都会消耗系统资源。在少量用户访问 Web 网站时问题还不大，但如果用户连接超过一定数量时，就会造成系统性能的下降，甚至死机。为了避免这种现象的发生，在完成数据库的操作后，应使用 mysql_close()函数关闭与 MySQL 服务器的连接，以节省系统资源。

语法格式如下：

```
mysql_close($Link);
```



说明

PHP 中与数据库的连接是非持久连接，系统会自动回收，一般不用设置关闭。但如果一次性返回的结果集比较大，或网站访问量比较多，则最好使用 mysql_close()函数手动进行释放。

有关 mysql_close()函数的使用可参考本章 18.2.3～18.2.7 节相关内容。

18.2　PHP 操作 MySQL 数据库的方法

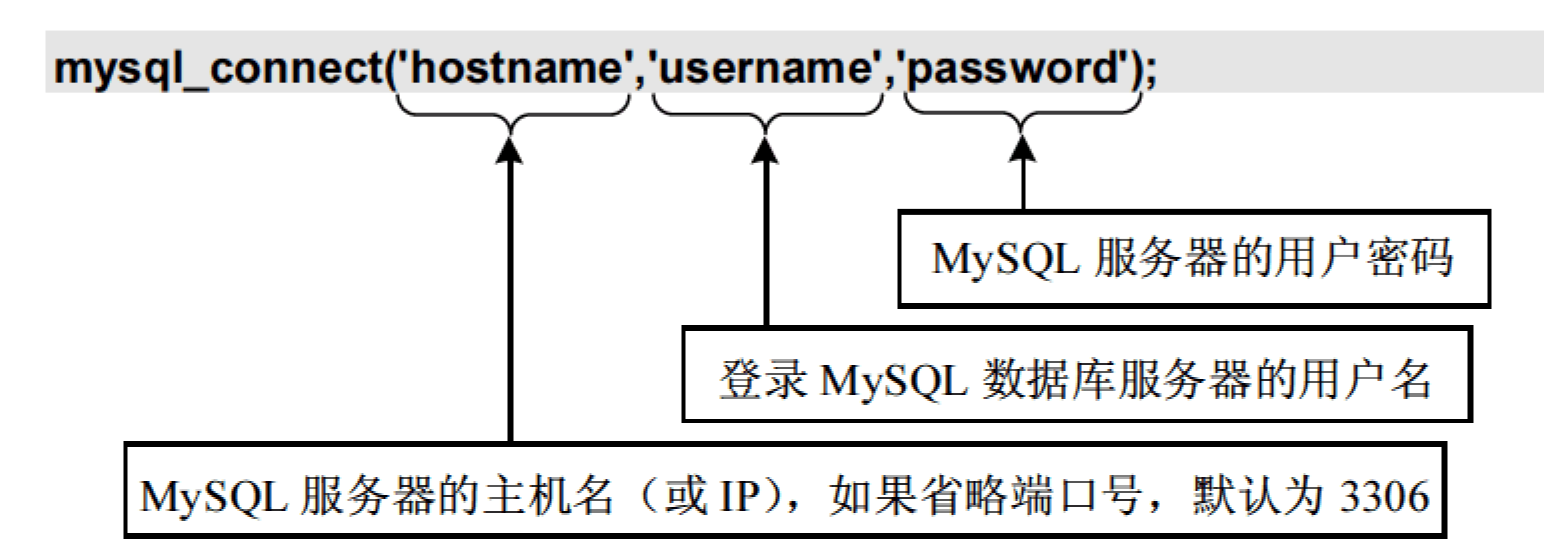
视频讲解：光盘\TM\lx\18\PHP 操作 MySQL 数据库的方法.exe

PHP 提供了大量的 MySQL 数据库函数，方便对 MySQL 数据库进行操作，使 Web 程序的开发更加简单灵活。

18.2.1　使用 mysql_connect()函数连接 MySQL 服务器

要操作 MySQL 数据库，首先必须与 MySQL 服务器建立连接。连接 MySQL 服务器的语句如下：

```
mysql_connect('hostname','username','password');
```



该函数的返回值用于表示这个数据库连接。如果连接成功，则函数返回一个资源，为以后执行 SQL 指令做准备。

【例 18.1】 使用 mysql_connect()函数在本地创建与 MySQL 服务器的连接，实例代码如下：**（实例位置：光盘\TM\sl\18\1）**

```
<?php
$link = mysql_connect("localhost", "root", "root") or die("不能连接到数据库服务器！可能是数据库服务器没有启动，或者用户名密码有误！".mysql_error());                    //连接 MySQL 服务器
if($link){
    echo "数据源连接成功!";
}
?>
```

（此段代码可以省略，如果运行后不出错，说明成功连接 MySQL 数据库服务器）

在上面的代码中，使用 mysql_connect()函数连接 MySQL 数据库服务器。从这个命令可以看到，可以指定非本机的机器名作为数据库服务器，这样就为数据的异地存放和数据库的安全隔离提供了保障。

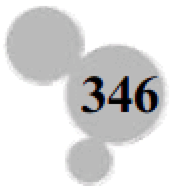
外界用户往往具有 WWW 服务器的直接访问权限，如果数据库系统直接放置在 WWW 服务器上，就会给 MySQL 数据库带来安全隐患，而如果为数据库系统安装防火墙，那么 PHP 可以通过局域网访问数据库，而局域网内部的计算机对外部是不可见的，这样就保证了数据库不受外来攻击。

为了方便查询因为连接问题而出现的错误，最好加上由 die()函数进行屏蔽的错误处理机制。本实例使用 mysql_error()函数提取 MySQL 函数的错误文本，如果没有出错，则返回空字符串，如果浏览器显示“**Warning**: mysql_connect()……”的字样时，说明是数据库连接的错误，这样就能迅速地发现错误位置，及时改正。

技巧

在网站开发时，错误是不可避免的，开发人员除了尽量避免出错外，及时准确地找到错误的起因也是必要的，这需要平时不断地积累经验。

在 IE 浏览器中输入地址，按 Enter 键，运行结果如图 18.2 所示。



技巧

在 mysql_connect()函数前面添加符号“@”，用于限制这个命令的出错信息的显示。如果函数调用出错，将执行 or 后面的语句。die()函数表示向用户输出引号中的内容后，程序终止执行。这样是为了防止数据库连接出错时，用户看到一堆莫名其妙的专业名词，而是提示定制的出错信息。但在调试时不要屏蔽出错信息，避免出错后难以找到问题。



图 18.2　创建与 MySQL 数据库的连接

注意

如果关闭 MySQL 服务器，则程序输出以下提示信息：

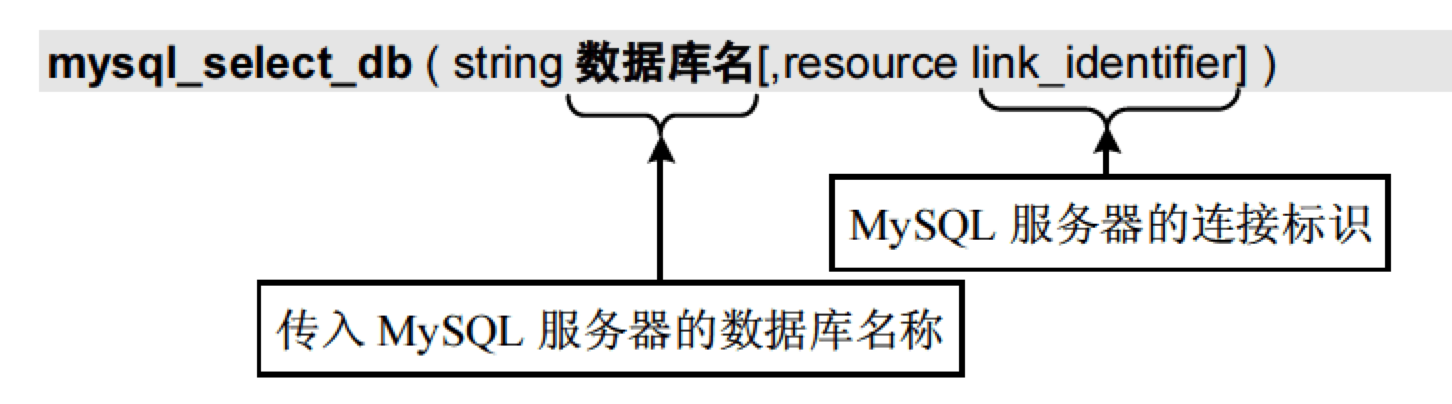
Can't connect to MySQL server on 'localhost' (10061)

18.2.2　使用 mysql_select_db()函数选择数据库文件

在连接到 MySQL 数据库服务器之后，接下来使用 mysql_select_db()函数选择数据库。

语法：

mysql_select_db (string **数据库名**[,resource link_identifier])



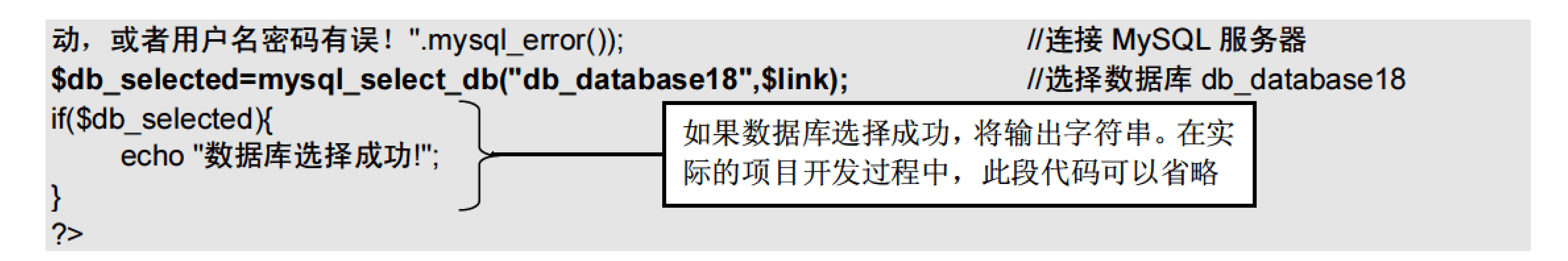
或

mysql_query("**use 数据库名**"[,resource link_identifier]);

如果没有指定连接标识符，则使用上一个打开的连接。如果没有打开的连接，本函数将无参数调用 mysql_connect()函数来尝试打开一个并使用。每个其后的 mysql_query()函数调用都会作用于活动数据库。

【例 18.2】 使用 mysql_select_db()函数连接 MySQL 数据库 db_database18，实例代码如下：（**实例位置：光盘\TM\sl\18\2**）

```
<?php
$link = mysql_connect("localhost", "root", "root") or die("不能连接到数据库服务器！可能是数据库服务器没有启
```



```
动，或者用户名密码有误！".mysql_error());                    //连接 MySQL 服务器
$db_selected=mysql_select_db("db_database18",$link);         //选择数据库 db_database18
if($db_selected){
    echo "数据库选择成功!";
}
?>
```


代码中加粗的部分也可以使用以下代码代替：

```
$db_selected=mysql_query("use db_database18",$link);         //选择数据库 db_database18
```

mysql_query()函数是查询指令的专用函数，所有的 SQL 语句都通过它执行，并返回结果集，关于 mysql_query()函数将在 18.2.3 节中进行详细介绍。

在 IE 浏览器中输入地址，按 Enter 键，运行结果如图 18.3 所示。

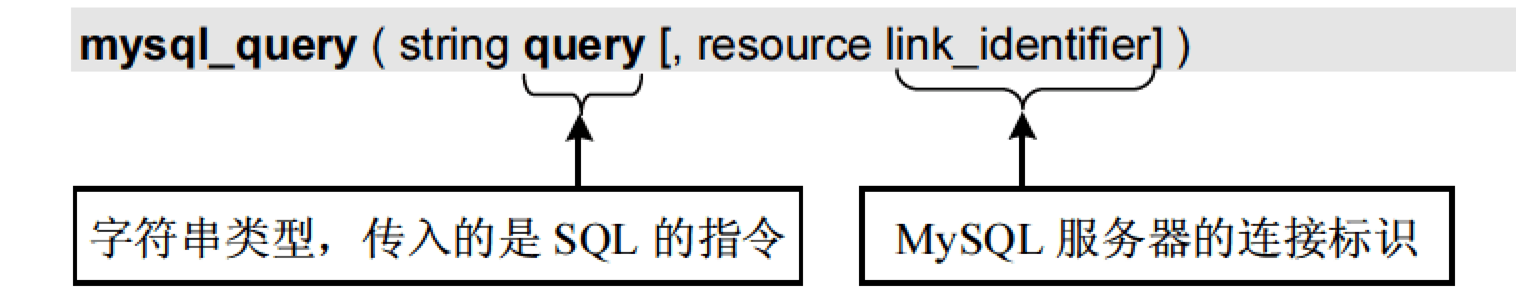


图 18.3　选择 MySQL 数据库

18.2.3　使用 mysql_query()函数执行 SQL 语句

要对数据库中的表进行操作，通常使用 mysql_query()函数执行 SQL 语句。

语法：



```
mysql_query ( string query [, resource link_identifier] )
```


mysql_query()函数是查询指令的专用函数，所有的 SQL 语句都通过它执行，并返回结果集。



注意

在 mysql_query()函数中执行的 SQL 语句不应以分号“;”结尾。

如果 SQL 语句是查询指令 select，成功则返回查询后的结果集，失败则返回 false；如果 SQL 语句是 insert、delete、update 等操作指令，成功则返回 true，失败则返回 false。

注意

mysql_unbuffered_query()函数向 MySQL 发送一条 SQL 查询语句，但不获取和缓存结果的行。它不像 mysql_query()函数那样自动获取并缓存结果集。一方面，这在处理很大的结果集时会节省可观的内存；另一方面，可以在获取第一行后立即对结果集进行操作，而不用等到整个 SQL 语句都执行完毕。

下面以会员信息表为例，举例说明常见的 SQL 语句的用法。

例如，执行一个添加会员记录的 SQL 语句的代码如下：

```
$result=mysql_query("insert into tb_member values('tm','111','tm@tmsoft.com')",$link);
```

例如，执行一个修改会员记录的 SQL 语句的代码如下：

```
$result=mysql_query("update tb_member set user='纯净水',pwd='1025' where user='tm'",$link);
```

例如，执行一个删除会员记录的 SQL 语句的代码如下：

```
$result=mysql_query("delete from tb_member where user='纯净水'",$link);
```

例如，执行一个查询会员记录的 SQL 语句的代码如下：

```
$result=mysql_query("select * from tb_member",$link);
```

例如，执行一个显示会员信息表结构的 SQL 语句的代码如下：

```
$result=mysql_query("DESC tb_member");
```

说明

在执行以上列出的 SQL 语句前，需要连接 MySQL 服务器和设置默认数据库 db_database18，其实现方法分别在 18.2.1 节和 18.2.2 节进行了详细介绍，这里不再赘述。

以上通过各个实例创建了 SQL 语句，并赋予变量$result。PHP 提供了一些函数来处理查询得到的结果**$result**，如 mysql_fetch_array()函数、mysql_fetch_object()函数和 mysql_fetch_row()函数等。为了便于读者理解这几个函数的具体应用以及各函数之间的区别，下面以查询图书信息为例进行分析。

18.2.4　使用 mysql_fetch_array()函数从数组结果集中获取信息

在 18.2.3 节讲解了使用 mysql_query()函数执行 SQL 语句，接下来使用 mysql_fetch_array()函数从数组结果集中获取信息。首先来看一下 mysql_fetch_array()函数的语法结构。

语法格式如下：

```
array mysql_fetch_array ( resource result [, int result_type] )
```

- ☑ result：资源类型的参数，要传入的是由 mysql_query()函数返回的数据指针。
- ☑ result_type：可选项，整数型参数，要传入的是 MYSQL_ASSOC（关联索引）、MYSQL_NUM（数字索引）、MYSQL_BOTH（同时包含关联和数字索引的数组） 3 种索引类型，默认值是 MYSQL_BOTH。

注意

本函数返回的字段名区分大小写，这是初学者最容易忽略的问题。

【例 18.3】 本实例实现一个图书信息检索的功能。首先，应用 mysql_query()函数执行 SQL 语句查询图书信息。然后，应用 mysql_fetch_array()函数获取查询结果。最后，使用 echo 语句输出数组结果集$info[]中的图书信息。**（实例位置：光盘\TM\sl\18\3）**

（1）创建 PHP 动态页，命名为 index.php。在 index.php 中，添加一个表单、一个文本框和一个提交按钮，代码如下：

```
<form name="myform" method="post" action="">
    <tr>
        <td width="605" height="51" bgcolor="#CC99FF"><div align="center">请输入图书名称
            <input name="txt_book" type="text" id="txt_book" size="25" >  
            <input type="submit" name="Submit" value="查询" >
        </div></td>
  </tr>
</form>
```

（2）连接到 MySQL 数据库服务器，选择数据库 db_database18，设置 MySQL 数据库的编码格式为 GB2312。其代码如下：

```
<?php
    $link=mysql_connect("localhost","root","root") or die("数据库连接失败".mysql_error());
    mysql_select_db("db_database18",$link);
    mysql_query("set names gb2312");        //设置 MySQL 数据库的编码格式为 GB2312 类型，以屏蔽乱码
?>
```

（3）使用 if 条件语句判断用户是否单击“查询”按钮，如果是则使用 POST 方法接收传递过来的图书名称信息，使用 mysql_query()函数执行 SQL 查询语句，该查询语句主要用来实现图书信息的模糊查询，查询结果被赋予变量$query。然后，使用 mysql_fetch_array()函数从数组结果集中获取信息。

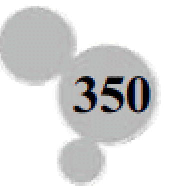
```
<?php
$sql=mysql_query("select * from tb_book");          //执行查询语句
$info=mysql_fetch_array($sql);                      //获取查询结果，返回值为数组
if ($_POST[Submit]=="查询"){                        //判断按钮 Submit 的值是否为查询
    $txt_book=$_POST[txt_book];                     //获取文本框提交的值
    $sql=mysql_query("select * from tb_book where bookname like '%". trim($txt_book)."%'");
                                                    //执行模糊查询
    $info=mysql_fetch_array($sql);                  //获取查询结果
}
?>
```

注意

本实例在实现模糊查询时，使用了通配符“%”。“%”表示零个或任意多个字符。

（4）使用 if 条件语句对结果集变量$info 进行判断，如果该值为假，则使用 echo 语句输出检索的图书信息不存在，代码如下：

```
<?php
if($info==false){                                   //如果检索的信息不存在，则输出相应的提示信息
```



```
        echo "<div align='center' style='color:#FF0000; font-size:12px'>对不起，您检索的图书信息不存在!</div>";
}
?>
```

（5）使用 do…while 循环语句以表格形式输出数组结果集$info[]中的图书信息。以字段的名称为索引，使用 echo 语句输出数组$info[]中的数据，代码如下：

```
<?php
do{                          //dowhile 循环
?>
    <tr align="left" bgcolor="#FFFFFF">
        <td height="20" align="center"><?php echo $info[id]; ?></td>
        <td > <?php echo $info[bookname]; ?></td>
        <td> <?php echo $info[issuDate]; ?></td>
        <td align="center"><?php echo $info[price]; ?></td>
        <td> <?php echo $info[maker]; ?></td>
        <td> <?php echo $info[publisher]; ?></td>
    </tr>
<?php
    }while($info=mysql_fetch_array($sql));        //判断循环语句
?>
```

运行本程序，默认将输出图书信息表中的全部图书信息，如图 18.4 所示。如果在文本框中输入欲检索的图书名称（由于支持模糊查询，因此可输入部分查询关键字），单击“查询”按钮，即可按条件检索指定的图书信息，并输出到浏览器，运行结果如图 18.5 所示。

请输入图书名称 项目开发全程实录 查询

编号	图书名称	出版时间	图书定价	作者	出版社
9	PHP网络编程自学手册	2008-03-01	52	明日科技	人民邮电出版社
5	PHP程序开发范例宝典	2007-06-30	89	明日科技	人民邮电出版社
6	PHP数据库系统开发完全手册	2007-06-01	52	明日科技	人民邮电出版社
7	PHP函数参考大全	2007-09-01	99	明日科技	人民邮电出版社
8	PHP项目开发全程实录	2008-04-01	65	明日科技	清华大学出版社

图 18.4　向文本框中输入查询关键字

请输入图书名称 查询

编号	图书名称	出版时间	图书定价	作者	出版社
8	PHP项目开发全程实录	2008-04-01	65	明日科技	清华大学出版社

图 18.5　使用 mysql_fetch_array()函数从数组结果集中获取图书信息

18.2.5　使用 mysql_fetch_object()函数从结果集中获取一行作为对象

使用 mysql_fetch_object()函数同样可以获取查询结果集中的数据。下面通过同一个实例的不同方

法来了解这两个函数在使用上的区别。首先来了解一下 mysql_fetch_object()函数。

语法格式如下：

```
object  mysql_fetch_object ( resource result )
```

mysql_fetch_object()函数和 mysql_fetch_array()函数类似，只有一点区别，即返回的是一个对象而不是数组，该函数只能通过字段名来访问数组。使用下面的格式获取结果集中行的元素。

```
$row->col_name                    //col_name 为列名，$row 代表结果集
```

例如，如果从某数据表中检索 id 和 name 值，可以用$row->id 和$row-> name 访问行中的元素值。



注意

本函数返回的字段名也是区分大小写的，这是初学者学习编程最容易忽视的问题。

【例 18.4】 本实例实现与例 18.3 相同的功能，不同的是本实例通过 mysql_fetch_object()函数获取结果集中的数据信息，然后使用 echo 语句从结果集中以对象“结果集->列名”的形式输出各字段所对应的图书信息。**（实例位置：光盘\TM\sl\18\4）**

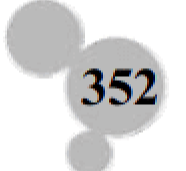
（1）创建项目、添加表单、连接 MySQL 服务器以及选择数据库的实现过程与实例 18.3 相同，这里不再赘述。

（2）与实例 18.3 不同的是，本实例使用 mysql_fetch_object()函数获取查询结果集中的数据，其返回值为一个对象，代码如下：

```
<?php
/***************************程序默认状态时，自动检索数据表中的全部数据信息*****************************/
$sql=mysql_query("select * from tb_book");
$info=mysql_fetch_object ($sql);
/**********************************************************************************************/
if ($_POST[Submit]=="查询"){               //判断查询按钮的值是否为“查询”
      $txt_book=$_POST[txt_book];          //获取文本框提交的查询关键字
      $sql=mysql_query("select * from tb_book where bookname like '%".trim($txt_book)."%'"); //执行模糊查询
      $info=mysql_fetch_object ($sql);   //获取查询结果，返回值为对象
}
?>
```

（3）使用 do…while 循环语句以对象的方式输出结果集中的图书信息，代码如下：

```
<?php
do{
?>
<tr align="left" bgcolor="#FFFFFF">
   <td height="20" align="center"><?php echo $info->id; ?></td>
   <td > <?php echo $info->bookname; ?></td>
   <td align="center"><?php echo $info->issuDate; ?></td>
   <td align="center"><?php echo $info->price; ?></td>
   <td align="center"> <?php echo $info->maker; ?></td>
   <td> <?php echo $info->publisher; ?></td>
```



```
</tr>
<?php
}while($info=mysql_fetch_object($sql));
?>
```

本实例的运行结果与实例 18.3 相同。

18.2.6　使用 mysql_fetch_row()函数逐行获取结果集中的每条记录

在 18.2.4 节和 18.2.5 节中讲解了使用 mysql_fetch_array()函数和 mysql_fetch_object()函数来获取结果集中的数据。本节向读者介绍第 3 种方法，使用 mysql_fetch_row()函数逐行获取结果集中的每条记录。首先来了解 mysql_fetch_row()函数。

语法格式如下：

```
array mysql_fetch_row ( resource result )
```

mysql_fetch_row()函数从和指定的结果标识关联的结果集中获取一行数据并作为数组返回，将此行赋予变量$row，每个结果的列存储在一个数组的单元中，偏移量从 0 开始，即以$row[0]的形式访问第一个元素（只有一个元素时也是如此），依次调用 mysql_fetch_row()函数将返回结果集中的下一行，直到没有更多行则返回 false。

注意

本函数返回的字段名区分字母大小写。

【例 18.5】 本实例实现与例 18.3 相同的功能，不同的是本实例通过 mysql_fetch_row()函数逐行获取结果集中的每条记录，然后使用 echo 语句从数组结果集中输出各字段所对应的图书信息。**（实例位置：光盘\TM\sl\18\5）**

（1）创建项目、添加表单、连接 MySQL 服务器以及选择数据库的实现过程与实例 18.3 相同，这里不再赘述。

（2）与实例 18.3 不同的是，本实例使用 mysql_fetch_row()函数逐行获取结果集中的记录，代码如下：

```
<?php
/**************************程序默认状态时，自动检索数据表中的全部数据信息************************/
$sql=mysql_query("select * from tb_book");
$row=mysql_fetch_row ($sql);
/*******************************************************************************************/
if ($_POST[Submit]=="查询"){
	$txt_book=$_POST[txt_book];	//获取文本框提交的查询关键字
	$sql=mysql_query("select * from tb_book where bookname like '%". trim($txt_book)."%'"); //执行模糊查询
	$row=mysql_fetch_row($sql);	//逐行获取查询结果，返回值为数组
}
?>
```

（3）使用 if 条件语句对结果集变量$info 进行判断，如果该值为假，则输出您检索的图书信息不存在；否则使用 do…while 循环语句以数组的方式输出结果集中的图书信息。其关键代码如下：

```
<?php
if($row==false){              //如果检索的信息不存在，则输出相应的提示信息
     echo "<div align='center' style='color:#FF0000; font-size:12px'>对不起，您检索的图书信息不存在!</div>";
}
?>
<?php
do{
?>
<tr align="left" bgcolor="#FFFFFF">
  <td height="20" align="center"><?php echo $row[0]; ?></td>
  <td > <?php echo $row[1]; ?></td>
  <td align="center"><?php echo $row[2]; ?></td>
  <td align="center"><?php echo $row[3]; ?></td>
  <td align="center"> <?php echo $row[4]; ?></td>
  <td> <?php echo $row[5]; ?></td>
</tr>
<?php
}while($row=mysql_fetch_row($sql));
?>
```

本实例的运行结果与实例 18.3 相同。

18.2.7 使用 mysql_num_rows()函数获取查询结果集中的记录数

要获取由 select 语句查询到的结果集中行的数目，则必须使用 mysql_num_rows()函数。

语法格式如下：

```
int mysql_num_rows ( resource result )
```

注意

使用 mysql_unbuffered_query()函数查询到的数据结果，无法使用 mysql_num_rows()函数来获取查询结果的记录数。

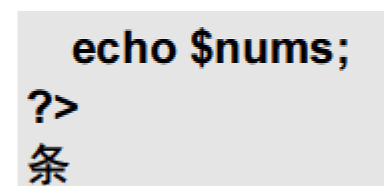
【例 18.6】 本实例沿用实例 18.3 的功能，在查询图书信息的同时，应用 mysql_num_rows()函数获取结果集中的记录数。**（实例位置：光盘\TM\sl\18\6）**

（1）本实例在实例 18.3 的基础上获取查询结果集中的记录数。

（2）在<body>标记内的任意位置使用 echo 语句输出由 mysql_num_rows()函数获取 SQL 查询语句结果集中的行数，代码如下：

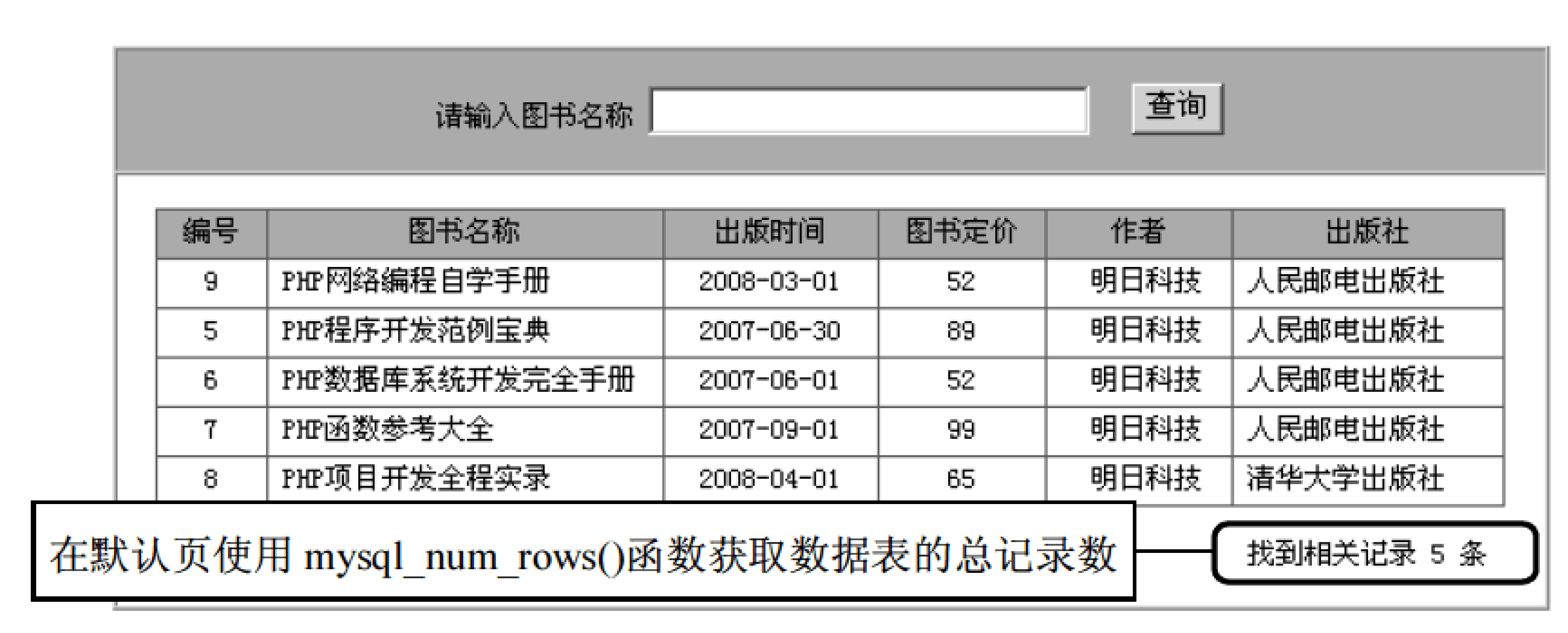
```
找到相关记录
<?php
  $nums=mysql_num_rows($sql);
```





```
    echo $nums;
?>
条
```

运行本程序，默认输出图书信息表中的全部图书信息，并自动汇总记录条数，如图 18.6 所示。在文本框中输入欲检索的图书名称，如“开发”（支持模糊查询，程序自动去除查询关键字左右空格），单击“查询”按钮，即可按条件检索指定的图书信息，并自动汇总检索到的记录条数，运行结果如图 18.7 所示。



编号	图书名称	出版时间	图书定价	作者	出版社
9	PHP网络编程自学手册	2008-03-01	52	明日科技	人民邮电出版社
5	PHP程序开发范例宝典	2007-06-30	89	明日科技	人民邮电出版社
6	PHP数据库系统开发完全手册	2007-06-01	52	明日科技	人民邮电出版社
7	PHP函数参考大全	2007-09-01	99	明日科技	人民邮电出版社
8	PHP项目开发全程实录	2008-04-01	65	明日科技	清华大学出版社

图 18.6　默认统计数据表中所有记录



编号	图书名称	出版时间	图书定价	作者	出版社
5	PHP程序开发范例宝典	2007-06-30	89	明日科技	人民邮电出版社
6	PHP数据库系统开发完全手册	2007-06-01	52	明日科技	人民邮电出版社
8	PHP项目开发全程实录	2008-04-01	65	明日科技	清华大学出版社

图 18.7　使用 mysql_num_rows()函数获取查询结果集中的记录数



技巧

如果要获取由 insert、update、delete 语句所影响到的数据行数，则必须使用 mysql_affected_ rows()函数来实现。

18.3　PHP 操作 MySQL 数据库

视频讲解：光盘\TM\lx\18\PHP 操作 MySQL 数据库.exe

PHP 数据库操作技术是 Web 开发过程中的核心技术，本节通过 PHP 和 MySQL 数据库实现一个公告信息管理，主要实现动态添加、修改、删除、查询和浏览公告信息。为读者学习 PHP+MySQL 数据库动态编程技术提供一个基本的思路。

18.3.1 使用 insert 语句动态添加公告信息

在实现动态添加公告信息前，首先需要明确数据表结构。创建数据表结构的方法有两种：一种是在命令提示符下创建，另一种是通过 phpMyAdmin 图形化管理工具创建。

☑ 在命令提示符下创建数据表结构

在命令提示符下，在 db_database18 数据库下创建 tb_affiche 公告信息表结构，代码如下：

```
CREATE TABLE 'tb_affiche' (
'id' INT( 4 ) NOT NULL AUTO_INCREMENT PRIMARY KEY ,
'title' VARCHAR( 200 ) CHARACTER SET gb2312 COLLATE gb2312_chinese_ci NOT NULL ,
'content' TEXT CHARACTER SET gb2312 COLLATE gb2312_chinese_ci NOT NULL ,
'createtime' DATETIME NOT NULL
) ENGINE = MYISAM ;
```

☑ 利用 phpMyAdmin 图形化管理工具创建数据表结构

在 phpMyAdmin 图形化管理工具中，选择 db_database18 数据库，然后按如图 18.8 所示的表结构创建数据表 tb_affiche。

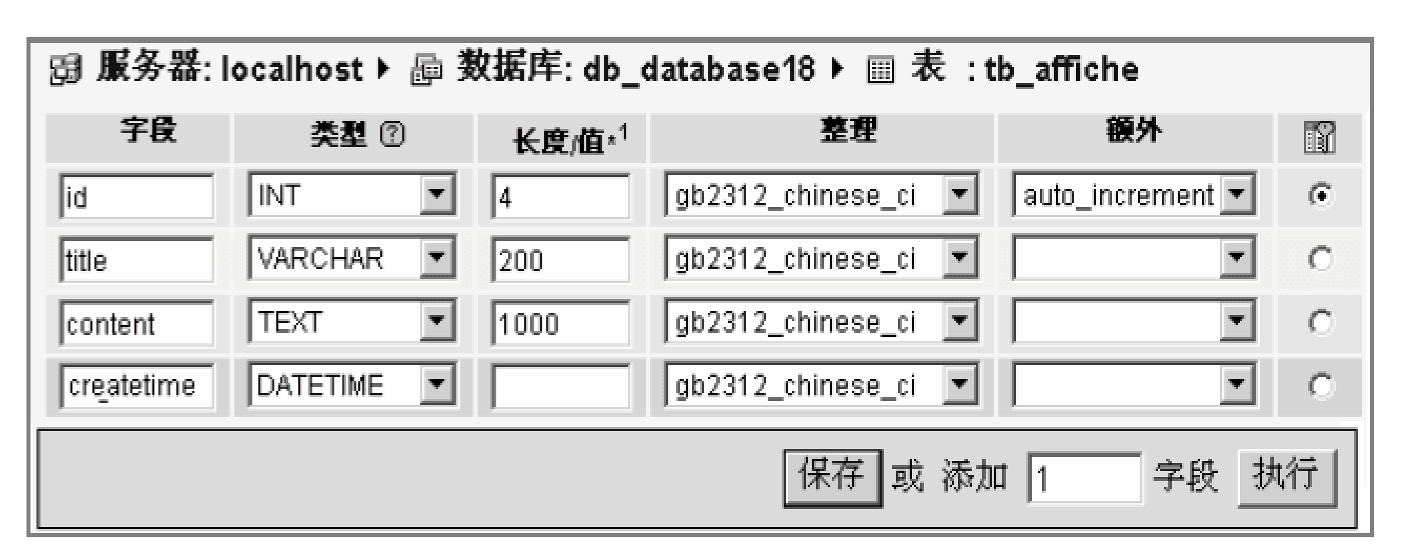


图 18.8　创建公告信息数据表

下面按照 PHP 访问 MySQL 数据库的一般步骤来讲解对数据进行动态添加的操作。

【例 18.7】 本实例主要使用 insert 语句动态地向数据库中添加公告信息，使用 mysql_query()函数执行 insert 语句，完成将数据动态添加到数据库的操作。（**实例位置：光盘\TM\sl\18\7**）

（1）创建 index.php 页面，完成页面布局。在添加公告信息的图片上添加热区，创建一个超链接，链接到 add_affiche.php 文件。其关键代码如下：

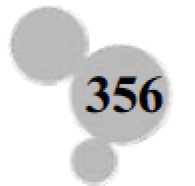
```
<img src="images/image_09.gif" width="202" height="310" border="0" usemap="#Map">
<map name="Map">
  <area shape="rect" coords="30,45,112,63" href=" add_affiche.php">
</map>
```



说明

在上面的代码中，coords="30,45,112,63"为热区的坐标。

（2）创建 add_affiche.php 页面，添加一个表单、一个文本框、一个编辑框、一个提交按钮和一个



重置按钮，设置表单的 action 属性值为 check_add_affiche.php，代码如下：

```
<form name="form1" method="post" action="check_add_affiche.php">
   <table width="520" height="212"   border="0" cellpadding="0" cellspacing="0" bgcolor="#FFFFFF">
     <tr>
        <td width="87" align="center">公告主题：</td>
        <td width="433" height="31"><input name="txt_titile" type="text" id="txt_titile" size="40"> * </td>
      </tr>
      <tr>
        <td height="124" align="center">公告内容：</td>
        <td><textarea name="txt_content" cols="50" rows="8" id="txt_content"></textarea></td>
      </tr>
      <tr>
        <td height="40" colspan="2" align="center">
       <input name="Submit" type="submit" class="btn_grey" value="保存"onClick="return check(form1)">
        <input type="reset" name="Submit2" value="重置">
        </td>
      </tr>
   </table>
</form>
```

通常，考虑到要严谨地添加公告信息，就不能过多地添加空信息。因此，在上面的代码中，在“保存”按钮的 onclick 事件下调用一个由 JavaScript 脚本自定义的 check()函数，用来限制表单信息不能为空，当用户单击“保存”按钮时，自动调用 check()函数，判断表单中提交的数据是否为空。check()函数的代码如下：

```
<script language="javascript">
function check(form){                                    //验证表单信息是否为空
      if(form.txt_title.value==""){                      //验证公告标题信息不能为空，否则弹出提示信息
          alert("请输入公告标题!");form.txt_title.focus();return false;
      }
      if(form.txt_content.value==""){                    //验证公告内容不能为空，否则弹出提示信息
          alert("请输入公告内容!");form.txt_content.focus();return false;
      }
form.submit();                                           //如果各控件不为空，提交表单信息到数据处理页
}
</script>
```

（3）创建 check_add_affiche.php 文件，对表单提交信息进行处理。首先，连接 MySQL 数据库服务器，选择数据库，设置数据库编码格式为 GB2312。然后，通过 POST 方法获取表单提交的数据。最后，定义 insert 语句将表单信息添加到数据表，通过 mysql_query()函数执行添加语句，完成公告信息的添加，弹出提示信息，并重新定位到 add_affiche.php 页面，代码如下：

```
<?php
  $conn=mysql_connect("localhost","root","root") or die("数据库服务器连接错误".mysql_error());
  mysql_select_db("db_database18",$conn) or die("数据库访问错误".mysql_error());
  mysql_query("set names gb2312");                       //选择编码格式为 GB2312
  $title=$_POST[txt_title];                              //获取公告标题信息
```

```
    $content=$_POST[txt_content];                          //获取公告内容
    $createtime=date("Y-m-d H:i:s");                       //获取系统的当前时间，“H”表示 24 小时制，必须大写
    /*************************应用 mysql_query()函数执行 insert …into 语句发送到服务器***********************/
    $sql=mysql_query("insert into tb_affiche(title,content,createtime)values('$title','$content','$createtime')");
    echo "<script>alert('公告信息添加成功!');window.location.href= add_affiche.php';</script>";
    mysql_free_result($sql);                               //关闭记录集
    mysql_close($conn);                                    //关闭 MySQL 数据库服务器
?>
```

在上面的代码中，date()函数用来获取系统的当前时间，内部的参数用来指定日期时间的格式，这里需要注意的是字母 H 要求大写，它代表时间采用 24 小时制计算。在公告信息添加成功后，使用 JavaScript 脚本弹出提示对话框，并在 JavaScript 脚本中使用 window.location.href='add_affiche.php'重新定位网页。

注意

在完成特定的功能后，要及时关闭记录集和 MySQL 服务器，以释放系统资源。

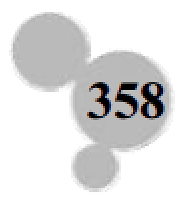
运行本实例，单击“添加公告信息”超链接，在页面中添加公告主题和公告内容，单击“保存”按钮，弹出“公告信息添加成功”提示信息，运行结果如图 18.9 所示。单击“确定”按钮，重新定位到公告信息添加页面。



图 18.9　添加公告信息页面的运行结果

18.3.2　使用 select 语句查询公告信息

实现添加公告信息后，即可对公告信息执行查询操作。下面在例 18.7 的基础上实现查询公告信息的功能。



【例 18.8】 本实例主要使用 select 语句动态检索数据库中的公告信息，使用 mysql_query()函数执行 select 查询语句，使用 mysql_fetch_object()函数获取查询结果，通过 do…while 循环语句输出查询结果。**（实例位置：光盘\TM\sl\18\8）**

程序开发步骤如下：

（1）在 index.php 页面中嵌入一个菜单导航页 menu.php。在 menu.php 页面中为菜单导航图片添加热区，链接到 search_affiche.php 页面。

（2）在 search_affiche.php 页面中，添加一个表单、一个文本框和一个提交按钮，代码如下：

```
<form name="form1" method="post" action="">
    查询关键字 
    <input name="txt_keyword" type="text" id="txt_keyword" size="40">
    <input type="submit" name="Submit" value="搜索" onClick="return check(form)">
</form>
```

为了防止用户搜索空信息，本程序在保存按钮的 onclick 事件下，调用一个由 JavaScript 脚本自定义的 check()函数，用来限制文本框信息不能为空，当用户单击“保存”按钮时，自动调用 check()函数，验证查询关键字是否为空。代码如下：

```
<script language="javascript">
function check(form){                                          //验证表单信息是否为空
     if(form.txt_keyword.value==""){                           //验证查询关键字不能为空，否则弹出提示信息
          alert("请输入查询关键字!");form.txt_keyword.focus();return false;
     }
form.submit();                                                 //如果各控件不为空，提交表单信息
}
</script>
```

在 search_affiche.php 页面中连接 MySQL 数据库服务器，选择数据库，设置数据库编码格式为 GB2312。通过 POST 方法获取表单提交的查询关键字，通过 mysql_query()函数执行模糊查询，通过 mysql_fetch_object()函数获取查询结果，通过 do…while 循环语句输出查询结果，最后关闭记录集和数据库，代码如下：

```
<?php
  $conn=mysql_connect("localhost","root","root") or die("数据库服务器连接错误".mysql_error());
  mysql_select_db("db_database18",$conn) or die("数据库访问错误".mysql_error());
  mysql_query("set names gb2312");                              //选择编码格式为 GB2312
  $keyword=$_POST[txt_keyword];                                 //获取查询关键字内容
  $sql=mysql_query("select * from tb_affiche where title like '%$keyword%' or content like '%$keyword%'");
  $row=mysql_fetch_object($sql);                                //获取查询结果集
  if(!$row){                                                    //如果未检索到信息资源，则弹出提示信息
     echo "<font color='red'>您搜索的信息不存在，请使用类似的关键字进行检索!</font>";
  }
  do{                                                           //使用 do…while 循环语句输出查询结果
  ?>
   <tr bgcolor="#FFFFFF">
     <td><?php echo $row->title;?></td>
     <td><?php echo $row->content;?></td>
```

```
</tr>
<?php
}while($row=mysql_fetch_object($sql));                        //do…while 循环语句结束
mysql_free_result($sql);                                      //关闭记录集
mysql_close($conn);                                           //关闭 MySQL 数据库服务器
?>
```

（3）在 IE 浏览器中输入地址，按 Enter 键，单击“查询公告信息”超链接，在页面中输入查询关键字，如“项目”，单击“搜索”按钮，即可输出检索到的公告信息资源，运行结果如图 18.10 所示。如果没有检索到相匹配的公告信息，则输出提示信息。

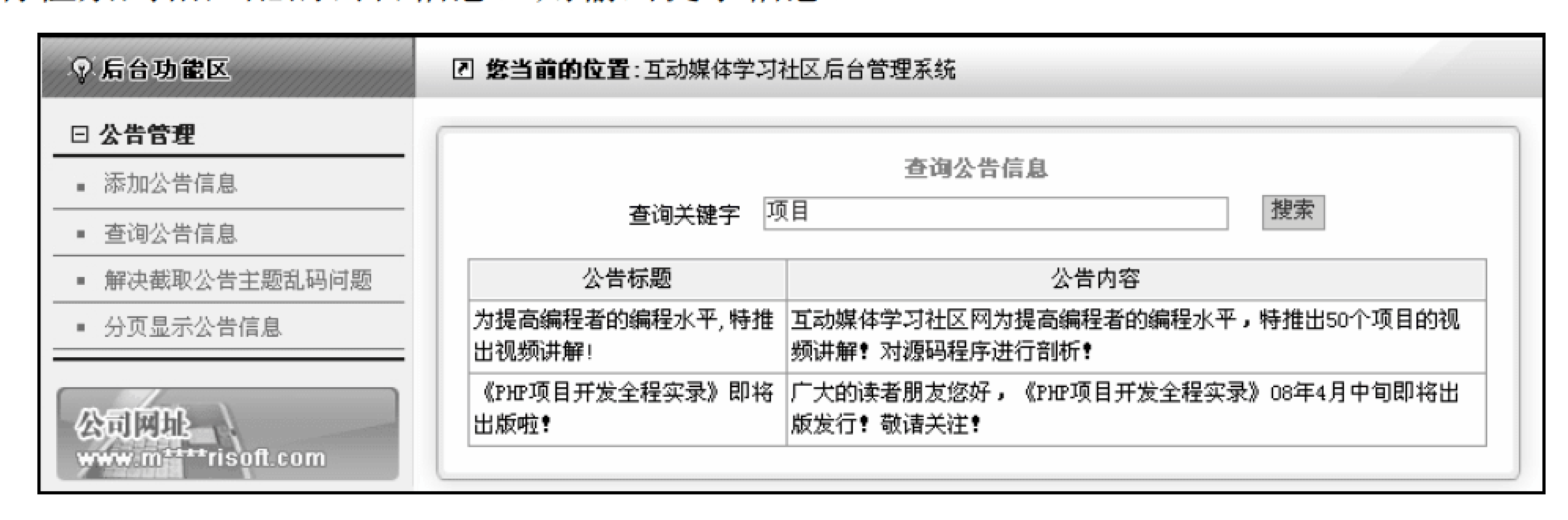


图 18.10　查询公告信息页面的运行结果

18.3.3　使用 update 语句动态编辑公告信息

公告信息不是一成不变的，可以对公告主题及内容进行编辑，下面在例 18.8 的基础上实现公告信息的编辑。

【例 18.9】 本实例主要使用 update 语句动态编辑数据库中的公告信息。**（实例位置：光盘\TM\sl\18\9）**

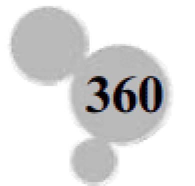
（1）在例 18.8 创建的菜单导航页 menu.php 中再添加一个热区，链接 update_affiche.php 文件。

（2）在 update_affiche.php 页面中，使用 select 语句检索出全部的公告信息，在通过表格输出公告信息时添加一列，在这个单元格中插入一个图标，并为这个图标设置超链接，链接到 modify.php 页面，并将公告的 ID 作为超链接的参数传递到 modify.php 页面中，其关键代码如下：

```
<a href="modify.php?id=<?php echo $row->id;?>">
    <img src="images/update.gif" width="20" height="18" border="0">
</a>
```

（3）创建 modify.php 公告信息编辑页面。首先完成与数据库的连接，然后根据超链接中传递的 ID 值从数据库中读取出指定的数据。然后，在该页面中添加一个表单、一个文本框、一个编辑框、一个隐藏域、一个提交按钮和一个重置按钮，设置表单的 action 属性值为 check_modify_ok.php。最后，将从数据库中读取出的数据在表单中输出。其关键代码如下：

```
<?php
$conn=mysql_connect("localhost","root","root") or die("数据库服务器连接错误".mysql_error());
mysql_select_db("db_database18",$conn) or die("数据库访问错误".mysql_error());
```



```
mysql_query("set names gb2312");                                    //设置编码格式
$id=$_GET[id];                                                      //使用 GET 方法接收欲编辑的公告 ID
$sql=mysql_query("select * from tb_affiche where id=$id");          //检索公告 ID 所对应的公告信息
$row=mysql_fetch_object($sql);                                      //获取结果集
?>
<form name="form1" method="post" action="check_modify_ok.php">
  <table width="520" height="212"   border="0" cellpadding="0" cellspacing="0" bgcolor="#FFFFFF">
     <tr>
        <td width="87" align="center">公告主题：</td>
        <td width="433" height="31"><input name="txt_title" type="text" id="txt_title" size="40" value="<?php
echo $row->title;?>"><input name="id" type="hidden"   value="<?php echo $row->id;?>"></td>
     </tr>
     <tr>
         <td height="124" align="center">公告内容：</td>
         <td><textarea name="txt_content"> <?php echo $row->content;?></textarea></td>
     </tr>
     <tr>
        <td height="40" colspan="2" align="center">
        <input name="Submit" type="submit" class="btn_grey" value="修改" onClick="return check(form1);">
        <input type="reset" name="Submit2" value="重置"></td>
      </tr>
   </table>
  </form>
```

（4）创建 check_modify_ok.php 表单提交数据处理页，根据表单隐藏域中传递的 ID 值，执行 update 更新语句，完成对指定公告信息的编辑，代码如下：

```
<?php
/**************连接数据源，读者可将此处封装成独立的页，然后使用 include 语句调用，以提高效率**************/
$conn=mysql_connect("localhost","root","root") or die("数据库服务器连接错误".mysql_error());
mysql_select_db("db_database18",$conn) or die("数据库访问错误".mysql_error());
mysql_query("set names gb2312");
/**********************************************************************************************/
$title=$_POST[txt_title];                                           //获取更改的公告主题
$content=$_POST[txt_content];                                       //获取更改的公告内容
$id=$_POST[id];                                                     //获取更改的公告 ID
//应用 mysql_query()函数向 MySQL 数据库服务器发送修改公告信息的 SQL 语句
$sql=mysql_query("update tb_affiche set title='$title',content='$content' where id=$id");
if($sql){
    echo "<script>alert('公告信息编辑成功！');history.back();window.location.href='modify.php?id=$id';</script>";
}else{
    echo "<script>alert('公告信息编辑失败！');history.back();window.location.href='modify.php?id=$id';</script>";
}
?>
```

运行本实例，在 index.php 页面中单击“编辑公告信息”超链接，进入到 update_affiche.php 页面，单击其中任意一条公告信息后的按钮，进入到公告信息编辑页，在该页面中完成对指定公告信息的编辑，最后单击“修改”按钮，完成指定公告信息的编辑操作，运行结果如图 18.11 所示。

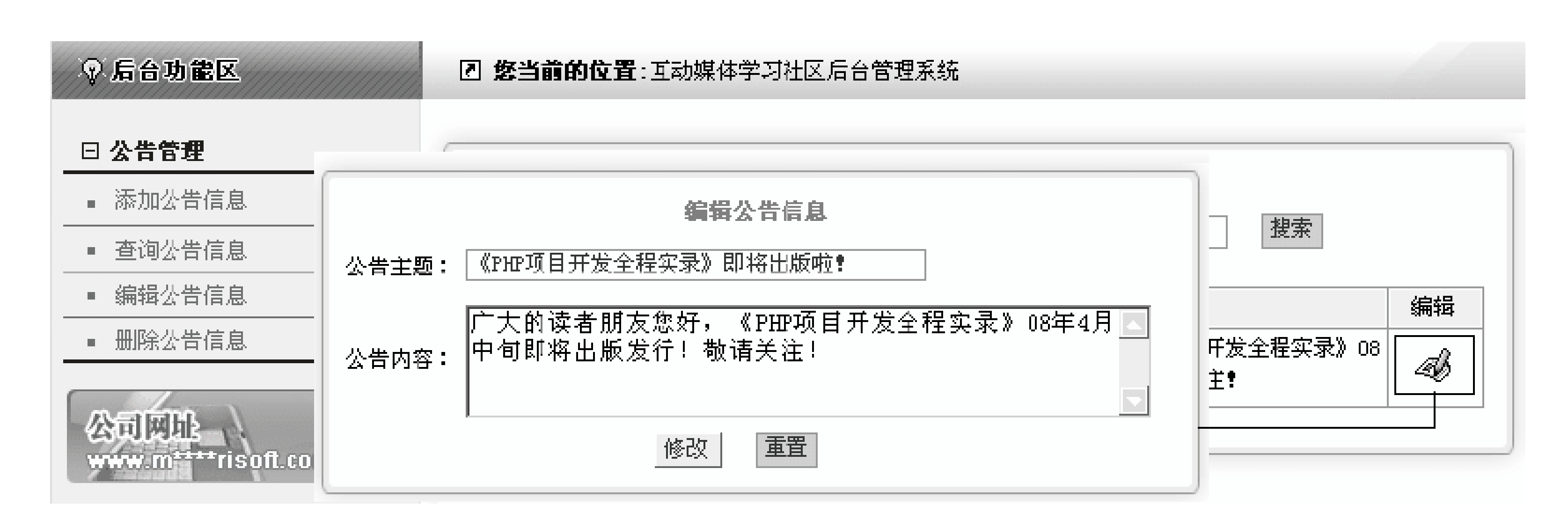


图 18.11　编辑公告页面的运行结果

18.3.4　使用 delete 语句动态删除公告信息

公告信息是用来发布网站或企业的最新信息，让浏览者了解网站的最新动态。因此，为了节省系统资源，可以定期地对公告主题和内容进行删除，下面在例 18.8 的基础上实现公告信息的删除。

【例 18.10】 本实例主要应用 delete 语句，根据指定的 ID，动态删除数据表中指定的公告信息。**（实例位置：光盘\TM\sl\18\10）**

（1）在菜单导航页 menu.php 中添加一个热区，链接到 delete_affiche.php 文件。

（2）创建 delete_affiche.php 页面，使用 Select 语句检索出全部的公告信息。在应用 do…while 循环语句通过表格输出公告信息时在表格中添加一列，并在单元格中插入删除图标，并将该图标链接到 check_del_ok.php 文件，将公告 ID 作为超链接的参数传递到 check_del_ok.php 文件中。其关键代码如下：

```
<a href="check_del_ok.php?id=<?php echo $row->id;?>">
     <img src="images/delete.gif" width="22" height="22" border="0">
</a>
```

（3）创建 check_del_ok.php 文件，根据超链接传递的公告信息的 ID 值，执行 delete 删除语句，删除数据表中指定的公告信息。最后使用 if…else 条件语句对 mysql_query()函数的返回值进行判断，并弹出相应的提示信息，代码如下：

```
<?php
/*******连接数据源，读者可将此处封装成独立的页，然后使用 include 语句调用，以提高效率********/
$conn=mysql_connect("localhost","root","root") or die("数据库服务器连接错误".mysql_error());
mysql_select_db("db_database18",$conn) or die("数据库访问错误".mysql_error());
mysql_query("set names gb2312");
/*****************************************************************************************/
$id=$_POST[id];                                                        //获取更改的公告 ID
//使用 mysql_query()函数向 MySQL 数据库服务器发送删除公告信息的 SQL 语句
$sql=mysql_query("update tb_affiche set title='$title',content='$content' where id=$id");
if($sql){
     echo "<script>alert('公告信息编辑成功！');history.back();window.location.href='modify.php?id=$id';</script>";
}else{
     echo "<script>alert('公告信息编辑失败！');history.back();window.location.href='modify.php?id=$id';</script>";
}
?>
```



技巧

在数据处理页 check_del_ok.php 中，由于该页都是动态代码，没有指定编码格式 GB2312 的编码类型，因此在 Dreamweaver 开发工具中打开该文件时，中文部分将会显示乱码。为了解决这一问题，读者可以在该页指定其编码格式，代码如下：

<meta http-equiv="Content-Type" content="text/html; charset=gb2312">

运行本实例，单击 index.php 页面中的“删除公告信息”超链接，在 delete_affiche.php 页面中，单击任意一条公告信息后的按钮，弹出删除公告信息提示，单击“确定”按钮，完成对指定公告信息的删除操作，运行结果如图 18.12 所示。

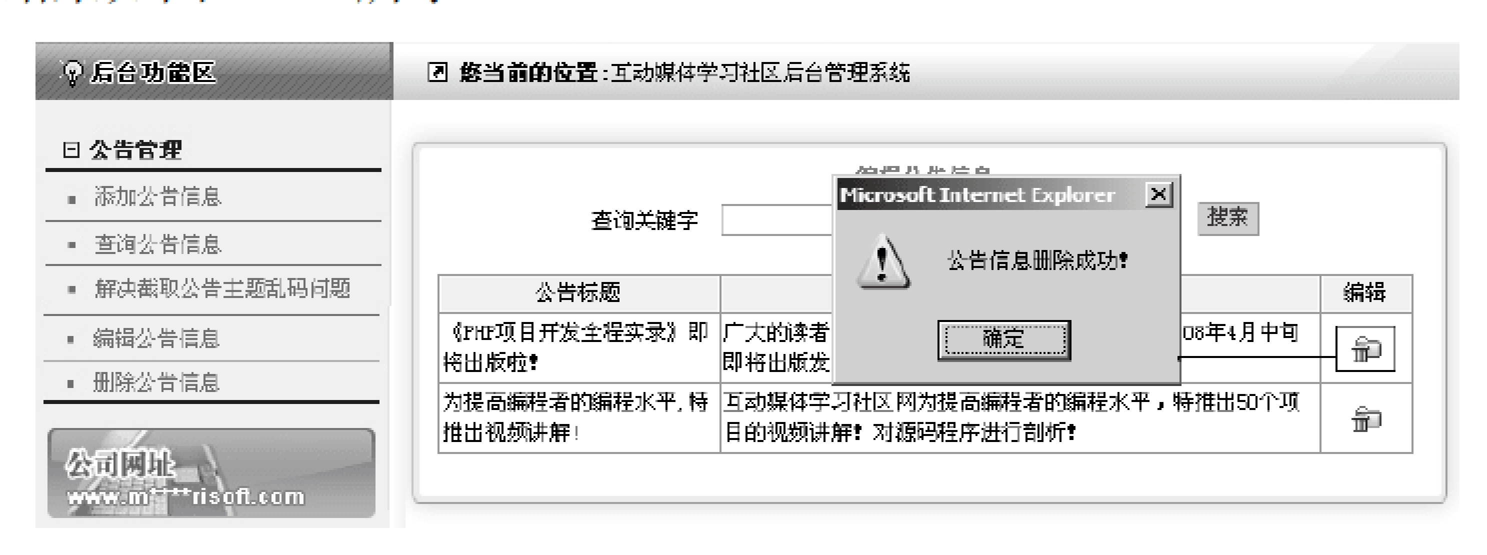


图 18.12　删除公告信息页面的运行结果

18.3.5　分页显示公告信息

在添加公告信息后，便可以对公告信息执行查询操作。为了更方便地浏览公告信息的内容，最好的方法就是通过分页来显示公告信息的内容。

【例 18.11】 本实例主要使用 select 语句动态检索数据库中的公告信息，并通过分页技术完成对数据库中公告信息的分页输出。**（实例位置：光盘\TM\sl\18\11）**

程序开发步骤如下：

（1）在菜单导航页 menu.php 中添加热区，链接到 page_affiche.php 文件。

（2）创建 page_affiche.php 页面，完成公告信息的分页输出，代码如下：

```
<table width="550" border="1" cellpadding="1" cellspacing="1" bordercolor="#FFFFFF" bgcolor="#999999">
 <tr align="center" bgcolor="#f0f0f0">
    <td width="221">公告标题</td>
    <td width="329">公告内容</td>
</tr>
<?php
  $conn=mysql_connect("localhost","root","root") or die("数据库服务器连接错误".mysql_error());
  mysql_select_db("db_database18",$conn) or die("数据库访问错误".mysql_error());
  /************这里必须指定数据库的编码方式为 GB2312，否则输出到浏览器中的公告信息为乱码***********/
  mysql_query("set names gb2312");
  /******************$page 为当前页，如果$page 为空，则初始化为 1*******************/
  if ($page==""){
```

```
    $page=1;}
    if (is_numeric($page)){                              //判断变量$page 是否为数字，如果是则返回 true
    $page_size=4;                                        //每页显示 4 条记录
    $query="select count(*) as total from tb_affiche  order by id desc";
    $result=mysql_query($query);                         //查询符合条件的记录总条数
    $message_count=mysql_result($result,0,"total");      //要显示的总记录数
    /****************根据记录总数除以每页显示的记录数求出所分的页数******************/
    $page_count=ceil($message_count/$page_size);
    $offset=($page-1)*$page_size;                        //计算下一页从第几条数据开始循环
    $sql=mysql_query("select * from tb_affiche order by id desc limit $offset, $page_size");
    $row=mysql_fetch_object($sql);                       //获取查询结果集
    if(!$row){                                           //如果未检索到信息资源，则输出提示信息
        echo "<font color='red'>暂无公告信息!</font>";

    }
    do{
    ?>
      <tr bgcolor="#FFFFFF">
        <td><?php echo $row->title;?></td>
        <td><?php echo $row->content;?></td>
      </tr>
<?php
    }while($row=mysql_fetch_object($sql));
}
?>
</table>
```

注意

do…while 循环是先执行{}中的代码段，然后判断 while 中的条件表达式是否成立，如果返回 true，则重复输出{}中的内容，否则结束循环，执行 while 下面的语句。while 循环先判断 while 中的表达式，当返回 true 时，再执行{}中的代码。do…while 循环比 while 循环多输出一次结果。

添加一个 1 行 1 列的表格，使用如下代码实现翻页功能：

```
<table width="550" border="0" cellspacing="0" cellpadding="0">
  <tr>
  <!--  翻页条 -->
    <td width="37%">  页次：<?php echo $page;?>/<?php echo $page_count;?>页 记
录：<?php echo $message_count;?>条  </td>
    <td width="63%" align="right">
    <?php
    /*  如果当前页不是首页  */
    if($page!=1){
    /*  显示“首页”超链接  */
    echo  "<a href=page_affiche.php?page=1>首页</a> ";
    /*  显示“上一页”超链接  */
    echo "<a href=page_affiche.php?page=".($page-1).">上一页</a> ";
```

```
        }
        /*   如果当前页不是尾页   */
        if($page<$page_count){
        /*   显示“下一页”超链接   */
        echo "<a href=page_affiche.php?page=".($page+1).">下一页</a> ";
        /*   显示“尾页”超链接   */
        echo   "<a href=page_affiche.php?page=".$page_count.">尾页</a>";
        }
        mysql_free_result($sql);                        //关闭记录集
        mysql_close($conn);                             //关闭 MySQL 数据库服务器
        ?>
       </tr>
    </table>
```

分页显示公告信息的运行结果如图 18.13 所示。



图 18.13　分页显示公告信息的运行结果

18.3.6　将数据库连接、操作、分页和字符串截取的方法封装到类中

可应用面向对象的方法，将数据库连接、操作、分页和字符串截取的方法都封装到类中，通过对类中方法的调用完成与数据库的连接、查询数据库中的数据、数据的分页显示和对字符串的截取。

【例 18.12】 本实例使用面向对象的方法，以网站公告信息的管理为背景，实现公告信息的添加、查找和分页输出的功能。这里主要讲解面向对象中类的创建和应用。**（实例位置：光盘\TM\sl\18\12）**

首先对数据库连接、操作、分页和字符串截取类进行封装。这里将创建的类都存储到 system.class.inc.php 文件中。

（1）创建数据库连接类 ConnDB，在该类中分别定义了 3 个方法，ConnDB()构造方法，为成员变量赋值；GetConnId()方法，通过 MySQL 数据库中的函数，完成与服务器和数据库的连接，设置编码格式，并返回连接标识；CloseConnId()方法，关闭数据库。ConnDB 类的代码如下：

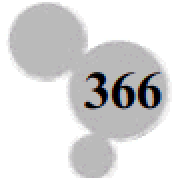
```
class ConnDB{
        var $dbtype;
```

```
    var $host;
    var $user;
    var $pwd;
    var $dbname;
    var $conn;
    function ConnDB($dbtype,$host,$user,$pwd,$dbname){    //构造方法，为成员变量赋值
        $this->dbtype=$dbtype;
    $this->host=$host;
        $this->user=$user;
        $this->pwd=$pwd;
        $this->dbname=$dbname;
    }
    function GetConnId(){                                  //实现与数据库的连接并返回连接对象
        $this->conn=mysql_connect($this->host,$this->user,$this->pwd) or die("数据库服务器连接错误".mysql_
error());
    mysql_select_db($this->dbname,$this->conn) or die("数据库访问错误".mysql_error());
    mysql_query("set names gb2312");                       //设置数据库的编码格式
    return $this->conn;                                    //返回连接对象
    }
    function CloseConnId(){                                //定义关闭数据库的方法
        $this->conn->Disconnect();                         //执行关闭的操作
    }
}
```

（2）创建数据库操作类 AdminDB，在该类中只有一个方法 ExecSQL()。通过 mysql_query()函数执行 SQL 语句。如果是 select 查询语句，则通过 mysql_fetch_array()函数获取查询结果，并返回数组；如果是 insert、update 或 delete 语句，执行成功则返回 true，否则返回 false。其代码如下：

```
class AdminDB{
    function ExecSQL($sqlstr,$conn){                   //定义方法，参数为 sql 语句和连接数据库返回的对象
        $sqltype=strtolower(substr(trim($sqlstr),0,6)); //截取 sql 中的前 6 个字符串，并转换成小写
        $rs=mysql_query($sqlstr);                      //执行 sql 语句
        if($sqltype=="select"){                        //判断如果 sql 语句的类型为 select
            $array=mysql_fetch_array($rs);             //执行该语句，获取查询结果
            if(count($array)==0 || $rs==false)         //判断语句是否执行成功
                return false;                          //如果查询结果为 0，或者执行失败，则返回 false
            else
                return $array;                         //否则返回查询结果的数组
        }elseif ($sqltype=="update" || $sqltype=="insert" || $sqltype=="delete"){
            //判断如果 sql 语句类型不为 select，则执行如下操作
            if($rs)
                return true;                           //执行成功返回 true
            else
                return false;                          //否则返回 false
        }
    }
}
```

（3）创建分页类 SepPage，在该类中包括 ShowDate()和 ShowPage()两个方法。ShowDate()方法实



现分页读取数据库中的数据，并将读取的数据存储到一个二维数组中，ShowPage()方法实现分页超链接的输出。SepPage 类的代码如下：

```
class SepPage{
	var $rs;
	var $pagesize;                                  //定义每页显示的记录数
	var $nowpage;                                   //当前页码
	var $array;
	var $conn;
	var $sqlstr;                                    //执行的 sql 语句
	var $total;
	var $pagecount;                                 //总的记录数
	function ShowDate($sqlstr,$conn,$pagesize,$nowpage){  //定义方法
		$arrays=array();
		$array_title=array();
		$array_content=array();
		if(!isset($nowpage) || $nowpage=="" || $nowpage==0)     //判断当前页变量值是否为空
			$this->nowpage=1;                          //定义当前页的值
		else
			$this->nowpage=$nowpage;                   //获取当前页的值
		$this->pagesize=$pagesize;                     //定义每页输出的记录数
		$this->conn=$conn;                             //连接数据库返回的标识
		$this->sqlstr=$sqlstr;                         //执行的查询语句
		$this->pagecount=$pagecount;                   //总的记录数
		$this->total=$total;                           //总的记录数
		$this->rs=mysql_query($this->sqlstr."limit ".
$this->pagesize*($this->nowpage-1).",$this->pagesize",$this->conn);        //执行查询语句
		$this->total=mysql_num_rows($this->rs);        //获取记录数
		if($this->total==0){                           //判断如果查询结果为 0，则输出如下内容
			return false;
		}else{                                         //否则
			if(($this->total % $this->pagesize)==0){    //判断如果总的记录数除以每页显示的记录数等于 0
				$this->pagecount=intval($this->total/$this->pagesize);     //则为变量 pagecount 赋值
			}else if($this->total<=$this->pagesize){
				$this->pagecount=1;             //如果查询结果小于等于每页记录数，那么为变量赋值为 1
			}else{
				$this->pagecount=ceil($this->total/$this->pagesize); //否则输出变量值
			}
			while($this->array=mysql_fetch_array($this->rs)){        //循环读取从数据库中获取的数据
				array_push($array_title,$this->array[title]);        //将数据添加到数组中
				array_push($array_content,$this->array[content]);    //将数据添加到数组中
			}
			array_push($arrays,$array_title,$array_content);         //将数据添加到数组中
			return $arrays;                                          //返回二维数组
		}
	}
	function ShowPage($contentname,$utits,$anothersearchstr,$class){
		$allrs=mysql_query($this->sqlstr,$this->conn);               //执行查询语句
		$record=mysql_num_rows($allrs);                              //获取记录数
```

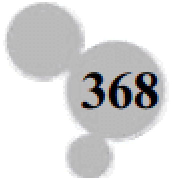
```
            $pagecount=ceil($record/$this->pagesize);                      //计算共有几页
            $str.="共有".$contentname." ".$record." ".$utits." 每页显示 ".$this->pagesize.
" ".$utits." 第 ".$this->nowpage." 页/共 ".$pagecount." 页";
            $str.="    ";
            $str.="<a href=".$_SERVER['PHP_SELF']."?page=1".$anothersearchstr." class=".$class.">首页</a>";
            $str.=" ";
            if(($this->nowpage-1)<=0){
                $str.="<a href=".$_SERVER['PHP_SELF']."?page=1".$anothersearchstr." class=".$class.">上一
页</a>";
            }else{
            $str.="<a href=".$_SERVER['PHP_SELF']."?page=".($this->nowpage-1).$anothersearchstr." class=". $class. ">
上一页</a>";
            }
            $str.=" ";
            if(($this->nowpage+1)>=$pagecount){
                $str.="<a href=".$_SERVER['PHP_SELF']."?page=".$pagecount.$anothersearchstr." class=".$class.">
下一页</a>";
            }else{
            $str.="<a href=".$_SERVER['PHP_SELF']."?page=".($this->nowpage+1).$anothersearchstr. "class=".
$class.">下一页</a>";
            }
            $str.=" ";
                $str.="<a href=".$_SERVER['PHP_SELF']."?page=".$pagecount.$anothersearchstr." class=".$class.">
尾页</a>";
            if(count($this->array)==0 || $this->rs==false)
                return "";
            else
               return $str;                                    //返回超链接字符串
        }
}
```

（4）创建字符串截取类 UseFun，有关字符串截取类的创建和使用可以参考本书第 14 章 14.4 节中的相关内容，这里不再赘述。

接下来完成类的实例化。这里将类的实例化操作都存储在 system.inc.php 文件中，在该文件中，完成了对数据库连接、操作、分页和字符串截取类的实例化，并分别返回实例化对象。system.inc.php 的代码如下：

```
<?php
require("system.class.inc.php");
//数据库连接类实例化，并传递参数，包括服务器、用户名、密码和数据库
$connobj=new ConnDB("mysql","localhost","root","root","db_database18");
$conn=$connobj->GetConnId();                    //连接数据库
$admindb=new AdminDB();                         //数据库操作类实例化
$seppage=new SepPage();                         //分页类实例化
$unhtml=new UseFun();                           //字符串截取类
?>
```

最后，在页面中调用类中的方法，完成与数据库的连接，实现数据的分页输出和对公告标题的截



取。page_affiche.php 的关键代码如下：

```
<table width="550" border="1" cellpadding="1" cellspacing="1" bordercolor="#FFFFFF" bgcolor="#999999">
    <tr align="center" bgcolor="#f0f0f0">
        <td width="221">公告标题</td>
        <td width="329">公告内容</td>
    </tr>
    <?php
    include_once("conn/system.inc.php");
    $myrow=$seppage->ShowDate("select * from tb_affiche order by id ",$conn,3,$_GET["page"]);
    if(!$myrow){
        echo "<font color='red'>暂无公告信息!</font>";
    }else{
        for($i=0;$i<count($myrow[0]);$i++){
    ?>
     <tr bgcolor="#FFFFFF">
        <td><?php echo $unhtml->chinesesubstr($myrow[0][$i],'0','30')."...";?></td>
        <td><?php echo $myrow[1][$i];?></td>
    </tr>
    <?php
        }
    }
    ?>
</table>
<table width="550" border="0" cellspacing="0" cellpadding="0">
    <tr><!--  翻页条  -->
        <td width="37%"><?php echo $seppage->ShowPage("公告","条","","a1");?></td>
    </tr>
</table>
```

本实例的运行结果如图 18.14 所示。



图 18.14　通过面向对象方法操作数据

18.4 小　　结

本章首先介绍了如何在 PHP 中连接并使用 MySQL 数据库，然后学习了如何使用 PHP 管理数据库中的数据，并进行增、删、改、查等操作。接着介绍了查询的实际应用，包括各种查询使用的方法和实现方式，最后讲解面向过程的分页显示技术和面向对象操作数据库的技术。通过本章的学习，相信读者可以熟练运用 PHP 函数操作 MySQL 数据库。

注意

在连接数据库时，一定要使用 mysql_query("set names gb2312")函数设置数据库的编码格式（其中的 gb2312 编码可以设置为其他编码）。通过该设置可以避免在输出中文字符串时出现乱码。

18.5 练习与实践

1．使用 PHP 操作 MySQL 数据库的方法开发一个动态的论坛，实现发布主题信息、查看主题信息、回复主题信息、删除主题及回复信息功能，并实现论坛主题信息的分页显示。**（答案位置：光盘\TM\sl\18\13）**

2．动态显示新闻信息，截取部分新闻主题字符串，屏蔽乱码。**（答案位置：光盘\TM\sl\18\14）**

第19章

ADODB 类库

（视频讲解：46 分钟）

PHP 为各种版本的数据库定义了庞大的函数群来支持，而且支持得很完整，那为什么还需要开发 ADODB 类库来对数据库进行操作呢？ADODB 的魅力到底在哪里，优势又是什么？如何使用 ADODB 进行开发？本章将一一进行讲解。

通过阅读本章，您可以：

- 了解 ADODB 类库
- 熟悉 ADODB 类库的下载、安装
- 掌握 ADODB 类库的常用函数
- 掌握通过 ADODB 类库创建数据库连接和操作类的方法
- 掌握通过 ADODB 类库创建分页类的方法
- 熟悉 ADODB 类库中事务的应用

19.1　ADODB 概述

视频讲解：光盘\TM\lx\19\ADODB 概述.exe

ADODB，全称是 Active Data Objects Data Base，它是存取数据库所使用到的一组函数。虽然 PHP 是建构 Web 系统强有力的工具，但是由于 PHP 存取函数一直没有标准化，不同数据库间的函数名称、参数差异很大，在更换数据库时，会带来大量的代码修复工作。这时，就需要一组函数库来隐藏不同数据库函数间的差异，让开发者可以很简单地去切换数据库，这就是 ADODB 类库。下面来看一下 ADODB 类库的优点：

☑ 安装简单，易学易用。提供了与微软的 ADODB 相似的语法功能。无论是初学者还是从其他语言转过来的开发人员（主要是 ASP），都可以很快上手。
☑ 以标准的 SQL 语句书写的程序在更换数据库时不需要改变源程序。
☑ 可以生成 Smarty 循环需要的二维数组，简化了 Smarty 开发。
☑ 支持缓存查询，最大可能地提高查询速度。

ADODB 也有缺点，即执行效率。ADODB 的类库非常庞大，仅是主执行类（adodb.inc.php）就有 120KB，所以在使用 ADODB 类库时要充分考虑到这一点。

ADODB 目前支持 MySQL、Oracle、Microsoft SQL Server、Sybase、PostgreSQL、FoxPro、Access、ADO 和 ODBC 等绝大多数的数据库，ADODB 所支持的常见数据库及参数如表 19.1 所示。

表 19.1　ADODB 支持的数据库

数据库名称	测试状态	资　料　库	RecordCount()支持	需要安装的驱动	操 作 系 统
access	B	Microsoft Access/Jet，需要建立一个 ODBC/DSN	Y/N	ODBC	Windows
db2	C	DB2，可以通过 ODBC 获得可信赖的运作效果	Y/N	DB2 CLI/ODBC Interface	UNIX Windows
vfp	A	Microsoft Visual FoxPro，需要建立一个 ODBC/ DSN	Y/N	ODBC	Windows
mssql	A	在日期格式上仍有一些问题	Y/N	Mssql Client	UNIX Windows
mysql	A	MySQL 支持交易处理	Y/N	MySQL Client	UNIX Windows
mysqlt 或 maxsql	A	MySQL 支持交易处理	Y/N	MySQL Client	UNIX Windows
oci8	A	Oracle 8/9，支持比 Oracle 驱动程式还多的功能。在连接之前，读者可能需要先配置好环境变数('ORACLE_HOME=...')	Y/N	Oracle Client	UNIX Windows
oci8po	A	Oracle 8/9 可携式驱动程序	Y/N	Oracle Client	UNIX Windows

续表

数据库名称	测试状态	资　料　库	RecordCount()支持	需要安装的驱动	操 作 系 统
odbc	A	标准 ODBC 用 PConnect('DSN','user','pwd').连接	? depends on database	ODBC	UNIX Windows
odbc_mssql	C	用 ODBC 连接 MSSQL	Y/N	ODBC	UNIX Windows
odbc_oracle	C	用 ODBC 连接 Oracle	Y/N	ODBC	UNIX Windows
postgres	C	PostgreSQL 不支援 LIMIT 指令	Y	PostgreSQL Client	UNIX Windows
sybase	C	Sybase	Y/N	Sybase Client	UNIX Windows

表 19.1 说明：

☑ A=已经经过很多人验证及测试，可靠度最高。

☑ B=已经测试并使用了，但可能仍有一些功能没有达成。

☑ C=使用者自行配置或试用的驱动程式，可能没有完全支持 ADODB 的功能。

☑ “RecordCount()支持”，指的是 RecordCount()函数是否会回传用 SELECT 指令取得的记录数（不支持传回-1）。Y/N 表示当全局变量$ADODB_COUNTER=true 时，会以模拟的方式取得，但如果估计到记录数会很大时，最好将该值设为 false，也就是关掉这个模拟功能，因为这会耗掉很多内存。这个变量在每次执行时都会检查，所以可以选择性地使用。

说明

要使用 ADODB 来操作数据库，首先就要获取和安装 ADODB。读者只要到网上下载 ADODB 类库包，将其解压到 Web 服务器目录下，然后在需要使用的位置调用 ADODB 中的文件即可。

注意

要使用 ADODB，使用的 PHP 必须是 4.01 以上的版本。

技巧

本书中使用的 ADODB 版本是 5.0。在本书中将下载的 ADODB 类库命名为 adodb5，存储在 tm\19 文件夹下。

19.2　使用 ADODB 操作 MySQL

视频讲解：光盘\TM\lx\19\使用 ADODB 操作 MySQL.exe

在详细介绍 ADODB 类库之前，首先来了解一下使用 ADODB 类库操作数据库的步骤。

【例 19.1】 本例使用 ADODB 类库中的常用方法，对表 tb_object 的数据进行读取。实例代码如下：（**实例位置：光盘\TM\sl\19\1**）

```
<?php
include_once ('../adodb5/adodb.inc.php');                                    //载入 adodb.ini.php 文件
$conn = ADONewConnection('mysql');                                           //连接 MySQL 数据库
$conn -> PConnect('localhost','root','root','db_database19');                //连接 db_database19 库
$conn -> execute('set names gb2312');                                        //设置编码
$rst = $conn -> Execute("select * from tb_object") or die('执行错误');       //执行 sql 语句
while(!$rst -> EOF){                                                         //配合 while 语句循环输出结果
    echo $rst -> fields['bigclass'].'<br>';
    $rst -> movenext();                                                      //指针下移
}
$rst -> close();
$conn -> close();
?>
```

结果为：图书 电器 服装 卫生洁具 生活用品 卫生洁具 卫生洁具

下面总结一下使用 ADODB 类库操作数据库的基本步骤：

（1）载入 adodb.inc.php 文件。要使用 ADODB 类库，首先要启动 ADODB，启动 ADODB 的方法就是载入 adodb.ini.php 文件。

（2）建立连接，使用 ADONewConnection()函数。

（3）连接数据库，使用 PConnect()函数。

（4）定义结果集的存取方式。

（5）执行 SQL 语句，使用 Execute 方法返回结果集。

（6）获取结果集。

（7）关闭连接。

19.3 ADODB 类库

视频讲解：光盘\TM\lx\19\ADODB 类库.exe

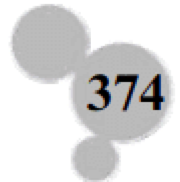
ADODB 是一个非常庞大的类库，仅一个 adodb.inc.php 文件大小就超过 100KB，这里主要介绍其中使用比较频繁的一些函数和变量。下面就以在 19.2 节中总结的基本步骤为顺序，对 ADODB 类库中的函数和变量进行分类讲解。

19.3.1 连接数据库的函数和方法

通过 ADODB 连接不同的数据库，可以使用以下 3 个函数：

（1）ADONewConnection()函数，连接数据库系统。

语法格式如下：



```
ADONewConnection($databaseType)
```

参数$databaseType 表示要连接的数据库系统的名称。例如“mysql”、“mssql”等。

（2）PConnect()函数，实现与数据库持久化连接。

语法格式如下：

```
PConnect($host,[$user],[$password],[$database])
```

参数说明如下。

- ☑ $host：数据库系统的服务器所在地址。如果是本机操作，格式为 localhost。
- ☑ $user：数据库用户名。
- ☑ $password：数据库密码。
- ☑ $database：使用到的数据库。

（3）Connect()函数，实现与数据库的非持久化连接。



说明

持久化连接和非持久化连接的区别在于：持久化连接不用每次都创建新连接，可以增加程序的执行速度。但有些数据库不支持，如果遇到不支持的数据库，可以使用 Connect()函数代替 PConnect()函数。

ADODB 支持多种数据库的连接，下面应用上述讲解的数据库连接函数，介绍一下如何实现与 MySQL、Microsoft SQL Server 和 Microsoft Access 数据库的连接。

（1）MySQL 数据库，连接 MySQL 数据库有两种不同的方法。

第一种是使用标准的连接字符串来连接，代码如下：

```
<?php
    include_once ('../adodb5/adodb.inc.php');                    //载入（include）adodb.inc.php 文件
    $conn = ADONewConnection('mysql');                           //建立连接
    $conn -> PConnect('localhost','root','root','db_database19'); //连接数据库
?>
```

其中 localhost、root、root 和 db_database19 分别表示要连接数据库服务器的地址、用户名、密码和数据库名称。

第二种是通过数据源名称（DSN）的方式进行连接，代码如下：

```
<?php
    include_once ('../adodb5/adodb.inc.php');                    //载入（include）adodb.inc.php 文件
    $conn = ADONewConnection('mysql://root:root@localhost/db_database19');   //连接数据库
?>
```

这两种不同的连接方法，实现的功能是相同的，都返回连接标识。

（2）Microsoft SQL Server 数据库，连接 Microsoft SQL Server 数据库使用 ODBC 的连接方法，代码如下：

```
<?php
	include_once ('../adodb5/adodb.inc.php');				//载入（include）adodb.inc.php 文件
	$conn = ADONewConnection('odbc_mssql');				//连接 SQL 数据库
	$conn-> PConnect("Driver={SQL Server};Server=localhost;Database=mydb;",'username','password');
?>
```

其中 localhost、username、password 和 mydb 分别表示要连接数据库服务器的地址、用户名、密码和数据库名称。

【例 19.2】 本实例通过 ADODB 使用 ODBC 方法连接 Microsoft SQL Server 数据库，查询 Microsoft SQL Server 系统库 master 中表 systypes，查询条件是 xtype 的值为 173，并输出查询结果。实例代码如下：**（实例位置：光盘\TM\sl\19\2）**

```
<?php

	include_once ('../adodb5/adodb.inc.php');							//载入（include）adodb.inc.php 文件
	$conn = ADONewConnection('odbc_mssql');							//连接 SQL 数据库
	$conn-> PConnect("Driver={SQL Server};Server=PKH;Database=master;",'sa','');
	$ADODB_FETCH_MODE = ADODB_FETCH_ASSOC;						//设置字段为结果集索引
	$sqlstr = 'select * from systypes where xtype = ?';				//创建 sql 语句
	$rst = $conn -> execute($sqlstr,'173') or die('connect error');	//执行 sql 语句，查询 xtype 等于 173 的数据
	while(!$rst -> EOF){										//如果没有错误，则配合 while 循环输出结果
		echo $rst -> fields['name'].'=>'.$rst -> fields['xtype'].'';	//输出查询结果
		$rst -> movenext();									//指针下移
	}
	$rst -> close();											//关闭连接
	$conn -> close();
?>
```

结果为：binary=>173

（3）Microsoft Access 数据库，连接 Access 数据库使用的是 ODBC 方法，代码如下：

```
<?php
	include_once ('../adodb5/adodb.inc.php');				//载入（include）adodb.inc.php 文件
	$conn = ADONewConnection('access');					//连接 Access 数据库
	$conn-> PConnect("Driver={Microsoft Access Driver (*.mdb)};Dbq=d:\\mydb.mdb;Uid=Admin;Pwd=;");
?>
```

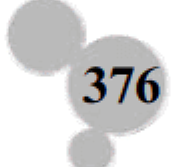
其中，d:\\mydb.mdb 表示数据库所在的文件名。

19.3.2 定义结果集的存取方式

对结果集存取方式的控制，使用的是 ADODB 类库中的公共变量。ADODB 类库中的公共变量不但可以对结果集存取方式进行控制，而且可以自动模拟 select 指令返回的记录总数和设置缓存目录。

（1）$ADODB_COUNTRECS 变量

当该变量为 true 时，RecordCount()函数会在数据库驱动不支持 select 指令返回的记录总数时进行自动模拟，来返回记录数，默认值为 true，每次进行查询操作时都会自动检查该变量的值。由于该功



能十分消耗内存，所以不建议使用。

（2）$ADODB_CACHE_DIR 变量

该变量设置缓存目录，一般在使用缓存函数时会用到该变量，例如 CacheExecute()函数。

（3）$ADODB_FETCH_MODE 变量

该变量决定结果集是以哪种方式来存取，包含 4 个值：

☑ define('ADODB_FETCH_DEFAULT',0)：默认值，在未设置该变量时使用。

注意

不同类型的数据库驱动默认值是不同的。

☑ define('ADODB_FETCH_NUM',1)：设置结果集以字段序号为索引进行存取。例如$rst -> fields[0]、$rst -> fields[1]。

☑ define('ADODB_FETCH_ASSOC',2)：设置结果集以字段名称为索引进行存取。例如$rst -> fields[bigclass]、$rst -> fields[id]。

☑ define('ADODB_FETCH_BOTH',3)：如果设置为该变量，那么同时支持上述两种方式。

注意

这里不建议使用变量 define('ADODB_FETCH_BOTH',3)，因为有很多的驱动并不支持。

【例 19.3】 本例中继续使用例 19.1 中的内容，将结果集以字段序号作为索引输出记录。实例代码如下：（**实例位置：光盘\TM\sl\19\3**）

```
<?php
include_once ('../adodb5/adodb.inc.php');                        //载入 adodb.ini.php 文件
$conn = ADONewConnection('mysql');                               //连接 MySQL 数据库
$conn -> PConnect('localhost','root','root','db_database19');    //连接 db_database19 库
$ADODB_FETCH_MODE = ADODB_FETCH_NUM;                             //设置结果集以字段序号为索引
$conn -> execute('set names gb2312');                            //设置编码
$rst = $conn -> Execute("select * from tb_object") or die('执行错误');   //执行 sql 语句
while(!$rst -> EOF){                                             //配合 while 语句循环输出结果
    echo $rst -> fields[1].'<br>';
    $rst -> movenext();                                          //指针下移
}
$rst -> close();
$conn -> close();
?>
```

结果为：图书　电器　服装　建筑耗材　卫生洁具　生

19.3.3　执行 SQL 语句

在完成与数据库的连接后，接下来要做的就是执行 SQL 语句，下面就来看一下在 ADODB 中通过

哪些函数来执行 SQL 语句。

（1）execute()函数。执行 SQL 语句，并返回一个结果集（ADORecordSet 对象）；失败则返回 false。语法格式如下：

```
execute($sql[,$inputarr=false])
```



注意

无论是执行 select 语句，还是 insert 或 update 语句，只要执行成功 execute 函数都会返回一个结果集。

参数说明如下。

- ☑ $sql：要执行的 SQL 语句。
- ☑ inputarr：用来作为传入的结合变量。

没有变量$inputarr 时，$sql 为普通的 SQL 语句，execute 的格式为：

```
$connect->Execute('select * from tb_object where id = 1');
```

当使用变量$inputarr 时，execute 的格式为：

```
$connect -> Execute('select * from tb_object where id = ?',array($vl));
```



说明

结合变量（Binding variables）的使用，可以加速 SQL 指令编译及读取的速度，产生较佳的效能。

【例 19.4】 本实例中，应用 execute()函数的第二个参数$inputarr，为 where 子句补充上完整的条件，查询出数据表中 ID 等于 2 的记录。实例代码如下：**（实例位置：光盘\TM\sl\19\4）**

```
<?php
    include_once 'conn/conn.php';                                  //连接数据库
    $ADODB_FETCH_MODE = ADODB_FETCH_ASSOC;                         //设置结果集以字段作为索引
    $sqlstr = 'select * from tb_object where id = ?';              //定义 sql 语句
    $rst = $conn -> execute($sqlstr,'2') or die('connect error');  //执行 sql 语句
    while(!$rst -> EOF){                                           //循环输出查询结果
        echo $rst -> fields['bigclass'].'=>'.$rst -> fields['d_time'].' ';
        $rst -> movenext();                                        //指针下移
    }
?>
```

结果为：电器=>2008-02-23 15:23:56



说明

在下面的实例中，为了代码整洁、便于数据库的管理，将连接数据库的操作定义到 conn.php 文件中，在使用时只要通过 include_once 语句进行调用即可。



（2）CacheExecute()函数。不但具有 execute()函数的功能，而且可以将查询结果保存到缓存中。如果以后还有相同的查询，就可以直接从缓存中获取。语法格式如下：

```
CacheExecute([$sec,]$sql[,$inputarr=false])
```

参数说明如下。

☑ $sec：查询结果集在缓存中被保存的时间。
☑ $sql：执行的 select 查询语句。这里只能是 select 语句。
☑ $inputarr：结合变量。

（3）SelectLimit($sql[,$numrows=-1][,$offset=-1][,$inputarr=false])函数。返回一个指定起始位置和记录数的结果集。语法格式如下：

```
SelectLimit($sql[,$numrows=-1][,$offset=-1][,$inputarr=false])
```

参数说明如下。

☑ $sql：要执行的 select 查询语句。
☑ $numrows：要查询的记录数。如果该值为-1，则函数会一直查询到最后一条记录。
☑ $offset：表示从第几条记录开始计算。
☑ $inputarr：结合变量。

说明

MySQL 数据库支持 limit 查询功能，但 Access 等数据库系统并不支持 limit。ADODB 考虑到这一点，模拟了一个和 limit 功能类似的函数。这里参数的位置和 MySQL 数据库中 limit 的参数位置不太一样，使用时需要注意。

【例 19.5】 本实例使用 selectlimit 函数查询数据，从第$offset 条记录开始查起，返回$numrows 条记录，最后输出数据。实例代码如下：**（实例位置：光盘\TM\sl\19\5）**

```
<?php
	include_once 'conn/conn.php';							//连接数据库
	$sql = 'select * from tb_object';						//定义查询语句
	$numrows = 2;									//设置查询的记录数
	$offset = 1;									//设置起始位置
	$rst = $conn -> selectlimit($sql,$numrows,$offset) or die('execute error：'.$conn -> ErrorMsg());//执行查询
	while(!$rst -> EOF){
		echo $rst -> fields['id'].'=>'.$rst -> fields['bigclass'];		//输出查询结果
		echo ' ';
		$rst -> movenext();							//指针下移
	}
?>
```

结果为：2=>电器　3=>服装

（4）CacheSelectLimit()函数。不但具有 SelectLimit()函数的功能，而且可以将查询结果保存到缓存当中。如果以后有相同的查询，将直接从缓存中获取。语法格式如下：

```
CacheSelectLimit([$sec,] $sql[, $numrows=-1][,$offset=-1][,$inputarr=false])
```

其参数含义同上。

（5）CachFlush()函数。清除所有 ADODB 数据库的缓存。

（6）GetUpdateSQL()函数。更新记录信息。语法格式如下：

```
GetUpdateSQL(&$rs, $arrFields, $forceUpdate=false)
```

参数说明如下。

☑ $rs：要更新的结果集。

☑ $arrFields：更新字段及更新内容。

☑ $forceUpdate：如果$forceUpdate 为 true，无论$arrFields 和$rs 的值是否相等都将更新。

【例 19.6】 本实例中，首先查找 ID 等于 1 的记录，然后创建要更新字段的数组，最后将该数组内容通过 GetUpdateSQL()函数更新到数据库中。实例代码如下：**（实例位置：光盘\TM\sl\19\6）**

```
<?php
    include_once 'conn/conn.php';                                          //载入数据库连接文件
    $sqlstr = 'select * from tb_object where id = 1';                      //声明 sql 语句
    $rst = $conn -> execute($sqlstr) or die('Error: '.$conn -> errorMsg());   //执行 sql 语句
    echo '修改前的数据，时间为：'.$rst -> fields['d_time'].'<br>';
    $fields = array();                                                     //定义空数组
    $fields['d_time'] = date("Y-m-d H:i:s");                               //定义时间
    $update = $conn -> GetUpdateSQL($rst,$fields) or die('update error：'.$conn -> errorMsg());//执行更新操作
    $conn -> execute($update);                                             //执行更新语句
    echo '修改后的数据，时间为：';
    $rst = $conn -> execute($sqlstr);                                      //执行 sql 语句
    echo $rst -> fields['d_time'];                                         //输出更新结果
?>
```

结果为：修改前的时间：2008-02-27 16:46:48　　修改后的时间：2009-12-17 09:15:20

（7）GetInsertSQL()函数。添加新记录。语法格式如下：

```
GetInsertSQL(&$rs, $arrFields)
```

参数说明如下。

☑ $rs：要添加记录的结果集。

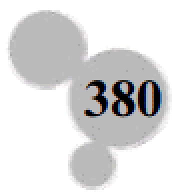
☑ $arrFields：新记录的字段值及对应字段名。

（8）DBDate()函数。实现不同数据库之间日期格式的转换。例如 MySQL 使用的时间格式为 Y-M-D，oracle 则为 D-M-Y。语法格式如下：

```
DBDate($date)
```

参数$date 表示要存储的时间。

（9）qstr()函数。使用引号来处理字符串。例如一个字符串包含了单引号“'”，在 MySQL 中可以直接使用“'”表示，但在其他的数据库中，则使用两个单引号“''”表示。使用 qstr()函数可以处理和解决这个问题。语法格式如下：



```
qstr($string)
```

参数$string 表示要被处理的字符串。

【例 19.7】 在本例中，使用 DBDate()函数和 qstr()函数向表 tb_object 中插入一条数据。实例代码如下：（**实例位置：光盘\TM\sl\19\7**）

```
<?php
    include_once 'conn/conn.php';                                              //载入数据库连接文件
    $bigclass = $conn -> qstr("建筑耗材");                                     //用 qstr()函数处理带引号的字串
    $d_time = $conn -> DBDate(date('Y-m-d H:i:s'));                            //用 DBDate()函数处理日期时间
    $sql = 'insert into tb_object(bigclass,d_time) value('.$bigclass.','.$d_time.')'; //生成 insert 语句
    $rst = $conn -> execute($sql) or die($conn -> errorMsg());                 //执行 insert 语句
    if($rst)
    $sqlstr = 'select * from tb_object where id order by id desc';             //声明 sql 语句
    $rst = $conn -> execute($sqlstr);                                          //执行 sql 语句
    echo $rst -> fields['bigclass']."----".$rst -> fields['d_time'];           //输出更新结果
?>
```

结果为：建筑耗材----2009-12-17 09:33:35

（10）Affected_rows()函数。获取最后更新或被删除的记录数。如果数据库不支持，返回 false。语法格式如下：

```
Affected_rows()
```

（11）Insert_ID()函数。返回最后插入的记录 ID 值。如果数据库不支持该函数，将返回 false。语法格式如下：

```
Insert_ID()
```

【例 19.8】 在本例中，仍然使用 DBDate()函数和 qstr()函数向表 tb_object 中插入一条数据，并通过 Affected_rows()函数返回最后添加的记录数，通过 Insert_ID()函数返回最后添加记录的 ID。实例代码如下：（**实例位置：光盘\TM\sl\19\8**）

```
<?php
    include_once 'conn/conn.php';                                              //载入数据库连接文件
    $bigclass = $conn -> qstr("建筑耗材");                                     //用 qstr()函数处理带引号的字串
    $d_time = $conn -> DBDate(date('Y-m-d H:i:s'));                            //用 DBDate()函数处理日期时间
    $sql = 'insert into tb_object(bigclass,d_time) value('.$bigclass.','.$d_time.')'; //生成 insert 语句
    $rst = $conn -> execute($sql) or die($conn -> errorMsg());                 //执行 insert 语句
    if($rst)
    $sqlstr = 'select * from tb_object where id order by id desc';             //声明 sql 语句
    $rst = $conn -> execute($sqlstr);                                          //执行 sql 语句
    echo $rst -> fields['bigclass']."----".$rst -> fields['d_time'];           //输出更新结果
    echo "<br>"."添加记录数：".$conn -> Affected_rows();                        //查看添加的记录数
    echo "<br>"."添加记录 ID：".$conn -> Insert_ID();                           //查看添加的记录 ID
?>
```

结果为：建筑耗材----2009-12-17 09:48:13　　添加记录数：15　添加记录 ID：59

19.3.4 控制结果集函数

当使用 execute()函数执行 SQL 指令时，会回传一个结果集。该结果集实际上是一个 ADORecordSet 对象。通过对该对象的控制，可以对结果进行各项操作。下面就来介绍一些常用的操作结果集的函数。

（1）fields 变量，保存当前指针所指向的记录。

（2）EOF 变量，记录当前指针是否指向最后一条记录。如果是最后一条记录，则被设定为 true；否则为 false。

（3）GetArray()函数，返回从当前指针指向的记录开始，到（$number_of_rows-1）行的全部记录的数组。语法格式如下：

```
GetArray([$number_of_rows])
```

参数$number_of_rows 是指定的记录行。如果没有给出，则一直到 EOF 才停止。

（4）UserDate()函数，将日期字符串$str 转换为变量$fmt 设置的日期格式。语法格式如下：

```
UserDate($str, [$fmt])
```

参数$str 是日期格式的字符串；$fmt 是要转换的日期格式。默认值为 Y-m-d。

（5）UserTimeStamp($str, [$fmt])函数，将时间字符串$str 转换为变量$fmt 设置的时间格式。语法格式如下：

```
UserTimeStamp($str, [$fmt])
```

参数$str 是时间字符串；$fmt 是时间格式设置字串。默认值为 Y-m-d H:i:s。

（6）MoveNext()函数，将 ADORecordSet（结果集）的指针下移一位。如果成功，则返回 true，否则返回 false。

（7）Move($to)函数，将 ADORecordSet（结果集）的指针移动到指定位置。如果$to 等于 0，则指针指向结果集的第一条数据；如果$to 的值大于结果集，则指针指向最后一条数据。

注意

这里变量$to 只能是绝对定位。

（8）MoveFirst()函数，指针移动到第一条数据。等同于 Move(0)。

（9）MoveLast()函数，指针移动到最后一条数据。等同于 Move(RecordCount() - 1)。

（10）FetchRow()函数，返回当前指针指向的记录的数组，如果是 EOF，回传 false。注意 FetchRow()不要和 MoveNext()混用。

（11）FetchField()函数，返回一个对象，包含着字段$column_number 的名称、空间长度等相关信息。语法格式如下：

```
FetchField($column_number)
```

参数$column_number 是要查看的字段名称。



（12）FetchNextObject()函数，返回当前指针所指向的记录的对象形式，并且指针自动下移一行。语法格式如下：

```
FetchNextObject([$toupper=true])
```

参数$toupper 如果等于 true，则字段名为大写形式。

（13）FieldCount()函数，返回结果集中的字段数。

（14）RecordCount()函数，返回结果集中的记录数。

（15）CurrentRow()函数，返回当前指针所指的记录序号。第一条记录用 0 来表示。

【例 19.9】 本实例通过 FieldCount()函数获取结果集记录数，然后通过 for 循环输出数据，并对每个字段类型进行判断，如果类型为“T”，则使用 DBDate()函数格式化输出时间，其他则直接输出。实例代码如下：**（实例位置：光盘\TM\sl\19\9）**

```
<?php
	include_once 'conn/conn.php';							//载入数据库连接文件
	$sqlstr = 'select * from tb_object where id = 1';				//sql 查询语句
	$rst = $conn -> execute($sqlstr) or die('error：	'.$conn -> errorMsg());	//执行查询语句
	if(false != $rst){								//如果有查询结果
		for($i = 0; $i < $rst -> FieldCount(); $i++){				//循环输出各个字段
			$fields = $rst -> FetchField($i);				//生成字段信息对象
			$type = $fields -> type;					//从对象中获取字段的类型信息
			echo '=>';
			echo $rst -> metaType($type,-1,$fields);			//输出字段标准类型
			if($rst -> metaType($type,-1,$fields) == "T")			//如果标准类型为“T”
				echo '('.$conn -> DBDate($rst -> fields[$i]).')';	//使用 DBDate 函数格式化时间
			else
				echo '('.$rst -> fields[$i].')';			//如果是其他类型，直接输出
		}
	}
?>
```

结果为：=>R(1)=>C(图书)=>T('2009-12-17 09:15:20')

19.3.5　生成 HTML 表格函数

rs2html()函数，返回一个 HTML 表格格式的结果集。语法格式如下：

```
rs2html($rst,[$table_attributes], [$col_titles])
```

参数说明如下。

☑ $rst：要返回的结果集。

☑ table_attributes：对表格的参数及属性设置。

☑ col_titles：对字段的重新命名。

注意

要使用这个函数，需要载入 tohtml.inc.php 文件。

【例 19.10】 本实例应用 rs2html()函数直接输出返回的结果集$rst，并设置表格的属性。实例代码如下：（**实例位置：光盘\TM\sl\19\10**）

```
<?php
    include_once 'conn/conn.php';                                           //载入数据库连接文件
    include_once '../adodb5/tohtml.inc.php';                                //载入 tohtml.inc.php 文件
    $rst = $conn -> execute('select * from tb_object');                     //返回查询结果集
    rs2html($rst,' width="350" border="1" cellpadding="1" cellspacing="1" bordercolor="#FFFFFF" bgcolor=
"#FF0000"',array('序号','类别','添加时间'));                                  //输出结果集
?>
```

运行结果如图 19.1 所示。



图 19.1　应用 rs2html()函数生成 HTML 表格

19.3.6　分页功能函数

在 ADODB 类库中，有属于 ADODB 自己的分页函数，下面分别进行介绍。

（1）PageExecute()函数，分页功能函数。语法格式如下：

```
PageExecute($sql, $nrows, $page)
```

参数说明如下。

☑　$sql：SQL 查询语句。

☑　$nrows：每页显示的记录数。

☑　$page：保存当前页数，默认为 1。

（2）CachePageExecute()函数，分页显示函数，同时将显示后的记录放到缓存中。在$sec秒内，如果是重复查看，将从缓存中读取数据。语法格式如下：

```
CachePageExecute($sec, $sql, $nrows, $page)
```

参数$sec 为数据在缓存中保留的时间，其他参数说明见 PageExecute()函数。

（3）AbsolutePage($page=-1)函数，返回当前的页数。需要和 PageExecute()配合使用，主要用于分页。

（4）AtFirstPage($status='')函数，如果当前页是第一页，返回 true。需要和 PageExecute()配合使用，用于分页。

（5）AtLastPage($status='')函数，如果当前是最后一页，返回 true。需要和 PageExecute()配合使用，用于分页。

（6）ADODB_Pager()函数，是 ADODB_Pager 类的构造函数，通过类中的 render()函数可以实现



一个小巧的分页操作。如果想要使用这个功能，需要载入 adodb_paper.inc.php 页。语法格式如下：

```
ADODB_Pager($conn, $sql, $id = 'adodb', $showPageLinks = false)
```

参数说明如下。

☑　$conn：数据库连接对象。

☑　$sql：执行的 SQL 语句。

☑　$id：每个分页的 ID 号。

☑　$showPageLinks：是否显示各个页的链接，默认为 false。

19.3.7　错误处理及调试

ADODB 的优势是可以让各种数据库函数统一起来，对相同的数据进行相同的操作。但在实际操作时，难免会出现各种各样的问题。下面就来学习几种错误处理及调试的方法。

1．debug

如果变量被设定为 true，当有输出操作时，同时也输出调试信息。

2．errorMsg()

函数作用：返回最后的状态或出错信息。即使没有错误发生，也会返回一个字符串，所以，一般只有在发生错误时（返回 false）才调用该函数。在启用 debug 变量后，只要有错误发生就会自动调用该函数。

【例 19.11】 在本例中，人为地将 SQL 语句中的 from 写成了 form，以查看输出的结果是什么。实例代码如下：**（实例位置：光盘\TM\sl\19\11）**

```
<?php
	include_once 'conn/conn.php';								//载入数据库连接文件
	$conn -> debug = true;									//开启调试
	$sql = 'select * form tb_object';							//sql 查询语句
	$rst = $conn -> execute($sql) or die($conn -> errorMsg());	//调用查询语句
?>
```

运行结果如图 19.2 所示。

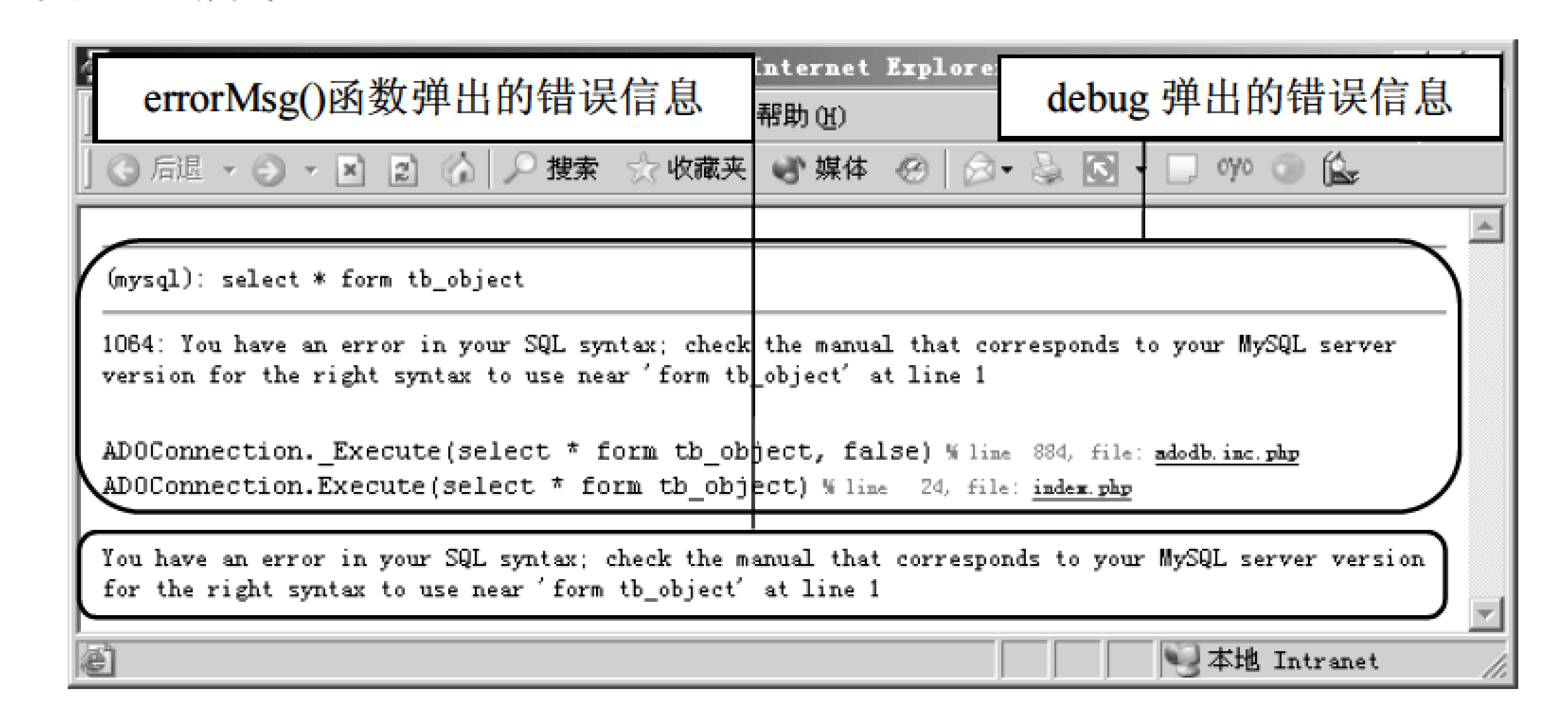


图 19.2　ADODB 调试

从图 19.2 中可以看到，debug 的错误提示中已经包含了 errorMsg()函数的错误信息。

19.4　ADODB 类库应用

视频讲解：光盘\TM\lx\19\ADODB 类库应用.exe

本节将结合实际，综合上面的函数介绍几个应用实例，使读者能够理解掌握，并熟练使用 ADODB 类库。

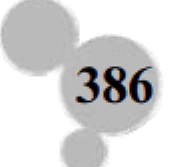
19.4.1　ADODB 连接、操作数据库类

【例 19.12】 在本实例中通过面向对象的方法，使用 ADODB 类库，创建数据库的连接类和数据库的操作类。通过类中的方法完成与数据库的连接，并读取数据库中的数据。**（实例位置：光盘\TM\sl\19\12）**

首先，在实例根目录下创建 conn 文件夹，编写 system.class.inc.php 文件。数据库连接类 ConnDB 和数据库操作类 AdminDB 都存储在这个文件中。

（1）通过数据库连接类 ConnDB 可以实现与不同类型数据库的连接，并返回连接对象。在 ConnDB 类中，ConnDB()构造方法为成员变量赋值；GetConnId()方法实现与不同类型数据库的连接；CloseConnId()方法关闭数据库。其代码如下：

```
class ConnDB{
	var $dbtype;
	var $host;
	var $user;
	var $pwd;
	var $dbname;
	var $debug;
	var $conn;
	function ConnDB($dbtype,$host,$user,$pwd,$dbname,$debug=false){	//构造方法，为成员变量赋值
		$this->dbtype=$dbtype;
		$this->host=$host;
		$this->user=$user;
		$this->pwd=$pwd;
		$this->dbname=$dbname;
		$this->debug=$debug;
	}
	function GetConnId(){										//实现与不同数据库的连接并返回连接对象
	require("../adodb5/adodb.inc.php");							//调用 ADODB 类库文件
		if($this->dbtype=="mysql" || $this->dbtype=="mssql"){	//判断成员变量传递的数据库类型
			if($this->dbtype=="mysql")							//判断如果是 MySQL 数据库
				$this->conn=NewADOConnection("mysql");			//执行与 MySQL 数据库的连接
			else
```



```
                    $this->conn=NewADOConnection("mssql");
                $this->conn->Connect($this->host,$this->user,$this->pwd,$this->dbname);
                                                        //数据库连接的用户名、密码
            }elseif($this->dbtype=="access"){           //判断如果使用的是 Access 数据库
                $this->conn=NewADOConnection("access");
                $this->conn->Connect("Driver={Microsoft Access Driver (*.mdb)};Dbq=".$this->dbname.";Uid=".$this->user.";
Pwd=".$this->pwd.";");                                  //执行连接 Access 数据库
            }
            $this->conn->Execute("set names gb2312");   //设置数据库的编码格式
            if($this->dbtype=="mysql")
                $this->conn->debug=$this->debug;
            return $this->conn;                         //返回连接对象
    }
    function CloseConnId(){                             //定义关闭数据库的方法
            $this->conn->Disconnect();                  //执行关闭的操作
    }
}
```

（2）通过数据库操作类 AdminDB 执行不同的 SQL 语句，实现对数据库中的数据进行操作。这里只有一个 ExecSQL()方法，其参数是执行的 SQL 语句和数据库的连接标识。代码如下：

```
class AdminDB{
    function ExecSQL($sqlstr,$conn){                    //定义方法，参数为 sql 语句和连接数据库返回的对象
        $sqltype=strtolower(substr(trim($sqlstr),0,6)); //截取 sql 中的前 6 个字符串，并转换成小写
        $rs=$conn->Execute($sqlstr);                    //执行 sql 语句
        if($sqltype=="select"){                         //判断如果 sql 语句的类型为 select
            $array=$rs->GetRows();                      //执行该语句，获取查询结果
            if(count($array)==0 || $rs==false)          //判断语句是否执行成功
                return false;                           //如果查询结果为 0，或者执行失败，则返回 false
            else
                return $array;                          //否则返回查询结果的数组
        }elseif ($sqltype=="update" || $sqltype=="insert" || $sqltype=="delete"){
            //判断如果 sql 语句类型不为 select，则执行如下操作
            if($rs)
                return true;                            //执行成功返回 true
            else
                return false;                           //否则返回 false
        }
    }
}
```

然后，实现对类的实例化操作，并定义返回的对象。类的实例化操作在 system.inc.php 文件中完成。其代码如下：

```
<?php
require("system.class.inc.php");                        //调用类文件
//数据库连接类实例化
$connobj=new ConnDB("mysql","localhost","root","root","db_database19",false);
$conn=$connobj->GetConnId();                            //执行数据库连接类中的方法，返回连接标识
//数据库操作类实例化
```

```
$admindb=new AdminDB();
?>
```

最后，调用类中的方法，完成与数据库的连接，并读取数据库中的数据，其关键代码如下：

```
<?php
	include_once 'conn/system.inc.php';				//连接数据库
	$sqlstr = 'select * from tb_object';				//定义 sql 语句
	$array = $admindb->ExecSQL($sqlstr,$conn);		//通过数据库操作类中的方法，执行查询
	for($i=0;$i<count($array);$i++){				//循环输出查询结果
?>
  <tr>
    <td align="center"><?php echo $array[$i][0];?></td>
    <td align="center"><?php echo $array[$i][1];?></td>
    <td align="center"><?php echo $array[$i][2];?></td>
  </tr>
<?php
  }
?>
```

运行结果如图 19.3 所示。



图 19.3　ADODB 连接、操作数据库类

19.4.2　ADODB 分页类

在 19.4.1 节中，将数据库的连接和操作方法都封装到了类中，这是一个非常好的方法，下面继续使用这个方法，应用 ADODB 类库中的分页函数，实现分页的功能，并且将分页技术也封装到一个类中。

【例 19.13】 本实例继续使用例 19.12 中定义的数据库连接和操作类，并且定义一个分页类，完成数据的分页输出。**（实例位置：光盘\TM\sl\19\13）**

这里有关数据库连接和操作的内容不再赘述，直接进行分页类的定义和应用。这里同样将分页类 SepPage 存储在 system.class.inc.php 文件中。在这个类中包括两个方法：ShowDate()方法，根据传递的参数，执行分页查询，完成对数据库中数据的分页读取；ShowPage()方法，应用 ADODB 类库中的分页函数创建分页超链接。SepPage 类的代码如下：

```
class SepPage{
	var $rs;
	var $pagesize;
```



```
	var $nowpage;
	var $array;
	var $conn;
	var $sqlstr;
	function ShowDate($sqlstr,$conn,$pagesize,$nowpage){		//定义方法
		if(!isset($nowpage) || $nowpage=="")			//判断变量值是否为空
			$this->nowpage=10;				//定义每页输出的记录数
		else
			$this->nowpage=$nowpage;
		$this->pagesize=$pagesize;				//定义每页输出的记录数
		$this->conn=$conn;					//连接数据库返回的标识
		$this->sqlstr=$sqlstr;				//执行的查询语句
		$this->rs=$this->conn->PageExecute($this->sqlstr,$this->pagesize,$this->nowpage);
		@$this->array=$this->rs->GetRows();			//获取记录数
			if(count($this->array)==0 || $this->rs==false)
				return false;
			else
				return $this->array;
	}
	function ShowPage($contentname,$utits,$anothersearchstr,$class){
		$allrs=$this->conn->Execute($this->sqlstr);		//执行查询语句
		$record=count($allrs->GetRows());			//统计记录总数
		$pagecount=ceil($record/$this->pagesize);		//计算共有几页
		$str.="共有".$contentname." ".$record." ".$utits." 每页显示 ".$this->pagesize."
 ".$utits." 第 ".$this->rs->AbsolutePage()." 页/共 ".$pagecount." 页";
		$str.="    ";
		if(!$this->rs->AtFirstPage())
			$str.="<a href=".$_SERVER['PHP_SELF']."?page=1".$anothersearchstr." class=".$class.">首页</a>";
		else
			$str.="<font color='#555555'>首页</font>";
		$str.=" ";
		if(!$this->rs->AtFirstPage())
			$str.="<a href=".$_SERVER['PHP_SELF']."?page=".($this->rs->AbsolutePage()-1).$anothersearchstr."
class=".$class.">上一页</a>";
		else
			$str.="<font color='#555555'>上一页</font>";
		$str.=" ";
		if(!$this->rs->AtLastPage())
			$str.="<a href=".$_SERVER['PHP_SELF']."?page=".($this->rs->AbsolutePage()+1).$anothersearchstr."
class=".$class.">下一页</a>";
		else
			$str.="<font color='#555555'>下一页</font>";
		$str.=" ";
		if(!$this->rs->AtLastPage())
			$str.="<a href=".$_SERVER['PHP_SELF']."?page=".$pagecount.$anothersearchstr." class=".$class.">
尾页</a>";
		else
			$str.="<font color='#555555'>尾页</font>";
		if(count($this->array)==0 || $this->rs==false)
```

```
                return "";
            else
                return $str;
        }
}
```

然后，依然是对类进行实例化，同样在 system.class.inc.php 文件中完成。其关键代码如下：

```
<?php
require("system.class.inc.php");                               //连接类文件
//数据库连接类实例化
$connobj=new ConnDB("mysql","localhost","root","root","db_database19",false);
$conn=$connobj->GetConnId();                                   //连接数据库，返回连接标识
$admindb=new AdminDB();                                        //数据库操作类实例化
$seppage=new SepPage();                                        //分页类实例化
?>
```

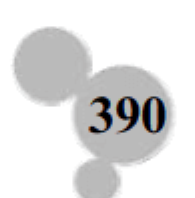
最后，调用类中方法完成数据库中数据的分页输出。其关键代码如下：

```
<?php
    include_once 'conn/system.inc.php';        //调用类中方法
    $array=$seppage->ShowDate("select * from tb_object",$conn,3,$_GET["page"]); //分页读取数据库中数据
    for($i=0;$i<count($array);$i++){            //循环输出数据库中数据
?>
  <tr>
    <td height="25" align="center" bgcolor="#FFFFFF"><?php echo $array[$i][0];?></td>
    <td align="center" bgcolor="#FFFFFF"><?php echo $array[$i][1];?></td>
    <td align="center" bgcolor="#FFFFFF"><?php echo $array[$i][2];?></td>
  </tr>
<?php
  }
?>
<tr>
  <td height="35" colspan="3" align="center" bgcolor="#FFFFFF"><?php echo $seppage->ShowPage("商品",
"类","","a1");?></td>
</tr>
```

运行结果如图 19.4 所示。

ID	类别	时间
55	建筑耗材	2009-12-17 09:41:26
31	卫生洁具	2008-02-26 19:55:30
17	生活用品	2008-02-26 14:48:28
共有商品 17 类 每页显示 3 类 第 2 页/共 6 页　首页 上一页 下一页 尾页		

图 19.4　通过分页类输出数据



19.5　小　　结

本章介绍了 ADODB 类库的优势、支持的数据库和类库中常用的函数，并对其中的部分函数进行了举例说明，而且还讲解了如何通过 ADODB 类库定义数据库连接、操作和分页类的方法。相信读者通过本章的学习，可以构建一个能随时更换并升级数据库的 Web 程序。

19.6　练习与实践

1．使用 BeginTrans()、CommitTrans()和 RollbackTrans() 3 个函数实现处理事务。**（答案位置：光盘\TM\sl\19\14）**

2．将 Access 数据库表中的数据导入到 MySQL 数据库中。**（答案位置：光盘\TM\sl\19\15）**

第20章

Zend Framework 框架

（视频讲解：1 小时 35 分钟）

本章将介绍基于 PHP 5 编写的、开源的 Web 开发框架 Zend Framework（简称 ZF），该框架采用 MVC 开发模式设计，并且每个组件又相对独立，这种松耦合结构又可以使开发者独立使用其中的每个组件。

使用 Zend Framework 可以简化开发流程、提高开发效率，开发者只要通过简单的调用就可以实现复杂的验证、分页、邮件发送、数据库 CRUD 操作等功能。

通过阅读本章，您可以：

- 掌握 Zend Framework 的环境搭建
- 配置站点初始参数并对站点进行布局
- 实现对数据库中信息缓存输出
- 实现对数据库中数据分页显示
- 掌握使用 Zend_Auth 对身份进行验证
- 掌握使用 Zend_Mail 发送邮件
- 熟悉 Zend Framework 开发项目流程

20.1　Zend Framework 概述

Zend Framework 项目由 Zend 公司主创，Google、Microsoft 和 StrikeIron 等公司也为框架的研发提供了技术支持和 Web 服务接口。Zend Framework 版本更新较快，所包含的组件也不断地增多，目前相对成熟，很多大型 Web 应用如 web968.com、zendchina.com 等都采用该框架开发。Zend Framework 所包含组件较多，常用组件如表 20.1 所示。

表 20.1　Zend Framework 中常用组件

组　件	功　能
Zend_Auth	主要用于认证，如用户注册，登录
Zend_Cache	为应用程序提供缓存服务
Zend_Config	简化应用程序中配置数据的使用
Zend_Controller	Zend Framework 的 MVC 体系的核心部分
Zend_Date	处理日期
Zend_Db	提供基于 PDO 的数据库操作方法
Zend_Debug	用于调试程序中表达式或变量的信息
Zend_Exception	Zend Framework 中的异常处理类，Zend Framework 抛出的所有异常都必须是它的子类的对象
Zend_Layout	实现经典的两步视图模型
Zend_Loader	提供动态加载文件和类功能
Zend_Mail	提供邮件发送功能

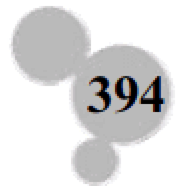
20.2　Zend Framework 环境搭建

视频讲解：光盘\TM\lx\20\Zend Framework 环境搭建.exe

Zend Framework 环境搭建主要包括 PHP 环境的配置和框架结构的建立。PHP 环境的配置包括启用 Apache 的 mod_rewrite 功能、开启 PHP 的 PDO_MySQL 扩展等。开发者可以使用手工和 Zend Studio 开发工具搭建 Zend Framework 的框架结构。

20.2.1　环境配置

使用 Zend Framework 框架进行项目开发，首先需要对 PHP 运行环境进行配置，从而使整个运行环境能够支持 Zend Framework 的各个组件和 MVC 运行机制。从 Zend Framework 的 MVC 运行机制可以知道，所有 HTTP 请求都需要访问 index.php 引导文件，而实现这个功能就需要使用 Apache 的 mod_rewrite 功能。由于 Zend Framework 的数据库访问是基于 PDO 的，所以需要开启 PHP 的 PDO_MySQL



扩展。综上可知，运行 Zend Framework 的基本环境配置操作包括开启 Apache 的 mod_rewrite 功能和启用 PHP 的 PDO_MySQL 扩展功能。

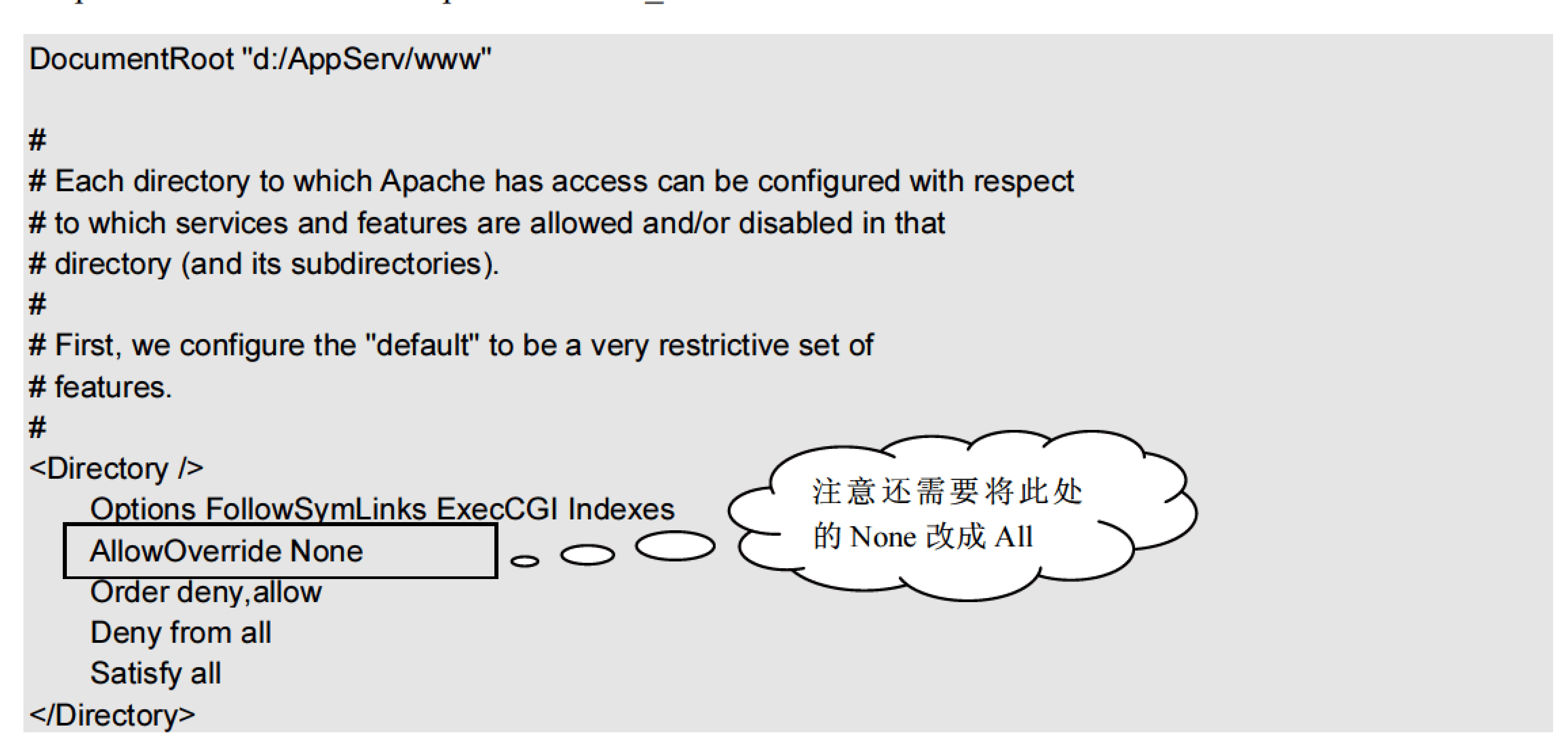
（1）开启 Apache 的 mod_rewrite 功能

进入 Apache 安装目录下的 conf 子目录，用“记事本”等文本编辑工具打开该目录下的 httpd.conf 文件，在该文件中搜索如下内容：

```
#LoadModule rewrite_module modules/mod_rewrite.so
```

找到上述配置信息后可以发现在其前有个“#”号，这是 httpd.conf 的注释标记符，将“#”号去掉则可以使 Apache 加载 mod_rewrite.so 模块。

使 Apache 加载 mod_rewrite.so 模块后还需要指定 rewrite 生效的目录，这里指定 Apache 的默认主目录为 rewrite 生效目录。在 httpd.conf 文件中找到“DocumentRoot”配置项，在该配置项后可以查看到“<Directory />”配置块，将该配置块中的“AllowOverride None”改为“AllowOverride All”，然后保存 httpd.conf 文件即可开启 Apache 的 mod_rewrite 功能。



```
DocumentRoot "d:/AppServ/www"

#
# Each directory to which Apache has access can be configured with respect
# to which services and features are allowed and/or disabled in that
# directory (and its subdirectories).
#
# First, we configure the "default" to be a very restrictive set of
# features.
#
<Directory />
    Options FollowSymLinks ExecCGI Indexes
    AllowOverride None
    Order deny,allow
    Deny from all
    Satisfy all
</Directory>
```

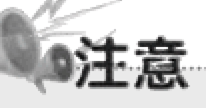

（2）启用 PHP 的 PDO_MySQL 扩展

进入系统的 WINDOWS 目录（这里以微软 Windows32 位操作系统为例），找到 php.ini 文件，同样用“记事本”等文本编辑工具打开该文件，并找到如下内容：

```
;extension=php_pdo.dll
......
;extension=php_pdo_mysql.dll
```

将这两处配置项前的“;”（php.ini 文件的注释标记）去掉，然后保存 php.ini 文件，即可开启 PHP 的 PDO_MySQL 扩展。开启上述扩展后还需要查看 PHP 安装目录下的 ext 子目录中是否存在 php_pdo_mysql.dll 文件和 php_pdo.dll 文件，如果不存在则需要下载这两个 dll 文件，然后复制到该子目录中。

完成以上配置后，需要重新启动 Apache 服务器才可以使配置生效。



注意

为了能够完全发挥 Zend Framework 各组件的功能，建议使用 PHP 5.2.6 或以上版本。

20.2.2　框架结构

Zend_Controller 是 Zend Framework 的 MVC（Model-View-Controller 的缩写）体系结构的核心部分，是一个用于分离应用逻辑和表现逻辑的设计模式，采用这种开发模式可以使数据库架构人员、前台设计人员和后台代码开发人员独立开来，可以有效地利用开发资源、提高开发效率，并且可以在很大程度上提高项目的可扩展性和可维护性。随着 Zend Framework 成熟化和所含组件的增多，其 MVC 框架结构也越来越人性化，并且 Zend Framework 框架设计上也十分灵活，如图 20.1 所示即为 Zend Framework 1.9 中典型的框架结构。

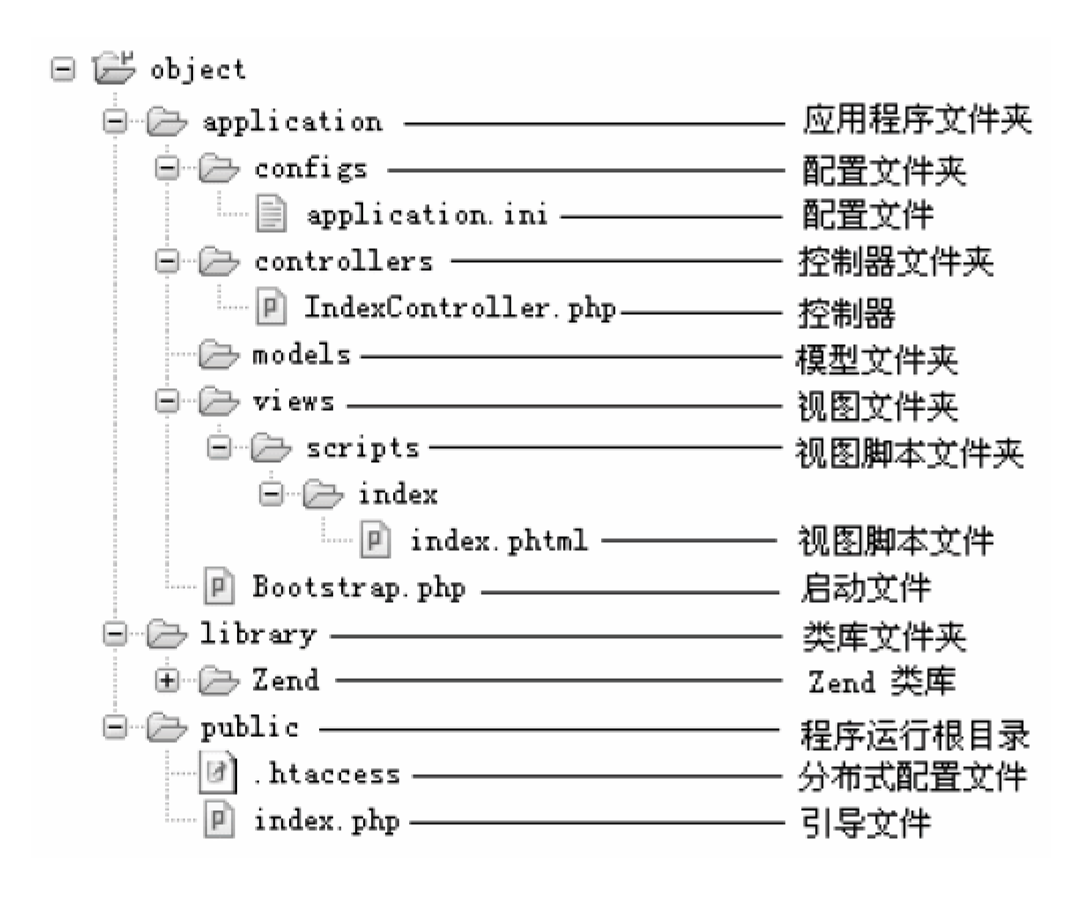


图 20.1　Zend Framework 的 MVC 体系结构图

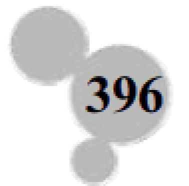
20.2.3　小试牛刀（Zend Framework 框架初始搭建）

前面对 Zend Framework 框架的环境配置和典型的框架结构进行了介绍，那么如何搭建 Zend Framework 的 MVC 体系结构呢？这也是初学者最为关心的知识点之一。

搭建 Zend Framework 的框架结构主要有两种方法，分别为手工搭建和使用开发工具搭建，手工搭建相对繁琐，对初学者来说可能较困难，不过还是建议初学者采用手工方法，这样可以提高对 Zend Framework 的 MVC 体系结构的理解程度，下面分别通过实例介绍这两种框架的搭建方法。

【例 20.1】 通过手工方法搭建 Zend Framework 的 MVC 框架结构，并简单地输出一段文字。（**实例位置：光盘\TM\sl\20\1**）

（1）创建工程 tm\sl\20\1，在 1 目录下创建程序的根目录 public，并在 public 目录下创建引导文件 index.php。代码如下：



```
<?php
set_include_path('../library'.PATH_SEPARATOR.get_include_path());                //设定路径
require_once 'Zend/Application.php';                                             //调用 Zend 类库
$application = new Zend_Application('project','../application/configs/application.ini');  //读取配置文件
$application->bootstrap()->run();                                                //运行方式　启动文件->运行
```

上述代码中，使用 set_include_path()方法，包含 library 目录，library 目录用于存放程序所依赖的类库、使程序在运行时能够直接找到这些类。Zend Framework 1.8 以前的版本中，所有项目的初始配置需要在引导文件 index.php 中手动编写，在 Zend Framework 1.8 以后的版本中，提供了 Zend_Application 组件，所有配置在扩展名为 ini 或 xml 的配置文件中指定，并通过 Zend_Application 组件进行解析。Zend_Application 组件的构造函数包含两个参数，其中第一个参数用于指定配置文件中环境参数的名称，第二个参数用于指定配置文件的位置。

实例化 Zend_Application 组件，然后调用该组件的 bootstrap()方法生成启动类对象，然后调用启动类的 run()方法运行程序。

说明

在 index.php 文件中，没有加入 PHP 的结束符“?>”，这是因为在文件结尾处它并不是必需的，而且这样可以避免产生一些难于调试的问题，还可以防止非法注入。

（2）在 public 目录下创建 URL 重写文件.htaccess，将所有不能映射到磁盘上已存在的文件的 URL 都用 index.php 来代替。其代码如下：

```
RewriteEngine on
RewriteRule !\.(js|ico|gif|jpg|png|css)$ index.php
```

如果在访问某些扩展名的文件时不转向到 index.php 引导文件，可以在 URL 重写中添加该扩展名，中间以“|”分隔。

（3）从步骤（1）可知，Zend Framework 的配置信息保存在一个扩展名为 ini 或 xml 的配置文件中，那么配置文件都对哪些参数进行了配置呢？主要包括启动类的路径、启动类的名称、错误级别、时区、控制器的目录、数据库连接参数及用户自定义的配置信息。本实例中 application.ini 配置文件的关键代码如下：

```
[project]                                                                      //头部标识
bootstrap.path = "../application/Bootstrap.php                                 //启动文件路径
bootstrap.class = "Bootstrap"                                                  //启动类名称
phpSettings.display_errors = 1                                                 //错误类型
phpSettings.date.timezone = "Asia/GMT-8"                                       //时间区域
resources.frontController.controllerDirectory = "../application/controllers"   //控制器路径
```

上述配置参数用于指定启动文件的路径以及启动文件的类名称，错误输出设置为 1，时间区域设置为中国地区时间，设置前端资源控制器路径。

（4）步骤（3）中，使用 application.ini 文件指定了启动类的路径和名称，本步将编写启动类，代码如下：

```
<?php
class Bootstrap extends Zend_Application_Bootstrap_Bootstrap
{
	public function _initAutoload()
	{
		$moduleAutoloader = new Zend_Application_Module_Autoloader(array('namespace' => '','basePath'
=> '../application'));
		return $moduleAutoloader;
	}
}
```

上述代码中创建了启动类，在_initAutoload()方法中实例化 Zend_Application_Module_Autoloader 对象并返回，这样的目的是使启动文件自动加载模块资源，设置的底层路径为应用程序文件夹 application，这样开发人员在编写程序时，就可以按照 Zend Framework 的目录结构引用类，而不需要使用 require()等包含函数包含要引用的类。

在启动类中以_init 开头的方法被当作资源方法自动运行，如果方法存在返回值，会被作为资源保存。

（5）完成以上操作后即可创建控制器。创建控制器之前，应该先理解 Zend Framework 是如何处理 HTTP 请求的。默认情况下，URL 的第一个部分会映射到一个控制器，第二个部分则映射到控制器类中的 Action（即控制器类内部的一个方法），如 URL 地址为“http://www.mingrbook.com/user/login”，其服务器路径为“/user/login”，则会映射到 user 控制器和 login 这个 Action。如果服务器路径不写 action，则会调用 index 这个 Action。如果服务器路径不写控制器，则会自动调用 index 控制器。

接下来，Zend Framework 的转发器会根据控制器的名称找到具体的控制器类。通常会把控制器名称加上“Controller”，并且控制器名称第一个字母大写，则“/user/login”转发到“UserController”控制器。

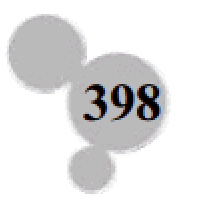
类似地，Action 会映射到控制器类中的一个类方法。默认情况下，会被转成小写字母，然后加上 Action。因此，“/user/login”则转发到 loginAction 这个动作。

下面创建一个 IndexController 控制器，并在其内容中创建一个名为 indexAction 的动作，代码如下：

```
<?php
class IndexController extends Zend_Controller_Action                    //创建 IndexController 控制器
{
	public function indexAction()                                         //创建 indexAction 动作
	{
		$this->view->assign('title','Zend Framework 环境初始搭建');        //为视图变量 title 初始化
		$this->view->assign('body','欢迎您搭建 Zend Framework 成功！');
	}
}
```

（6）创建完成控制器后还需要创建视图，默认视图保存路径为与 Controllers 同级目录下的 views 目录的 scripts 子目录中，视图文件则为 scripts 目录下的 index/index.phtml。index.phtml 视图文件的关键代码如下：

```
<?php echo $this->escape($this->title); ?>                              //输出视图变量
<?php echo $this->escape($this->body); ?>
```



（7）打开浏览器，在地址栏中输入如下 URL 地址：

http://127.0.0.1/TM/sl/20/1/public/

http://127.0.0.1/TM/sl/20/1/public/index

http://localhost/TM/sl/20/1/public/index/index

可以发现通过上述地址访问，运行效果均如图 20.2 所示，根据步骤（5）的规则，访问地址应该是“http://localhost/TM/sl/ 20/1/public/index/index”，那么为什么会可以省略掉控制器名或动作名呢？这是因为 Zend Framework 中如果省略了控制器名或动作名，则默认为 index。



图 20.2　Zend Framework 环境初始搭建

【例 20.2】 使用 Zend Studio 开发工具搭建 Zend Framework 的 MVC 框架结构。（**实例位置：光盘\TM\sl\20\2**）

Zend Studio 是由 Zend 公司开发，用于进行 PHP 开发的工具，包括了编辑、调试、配置 PHP 程序所需要的客户及服务器组件。尤其是强大的代码提示功能和齐全的代码调试功能，使开发人员的代码编写工作更加游刃有余。可以从 Zend 官方网站下载到 Zend Studio 的最新版本，目前下载地址为：http://www.zend.com/store/products/zend-studio.php。下面介绍使用 Zend Studio 开发工具搭建 Zend Framework 的 MVC 框架结构。

（1）启动 Zend Studio，选择 File/New/Zend Framework Project 命令，如图 20.3 所示。

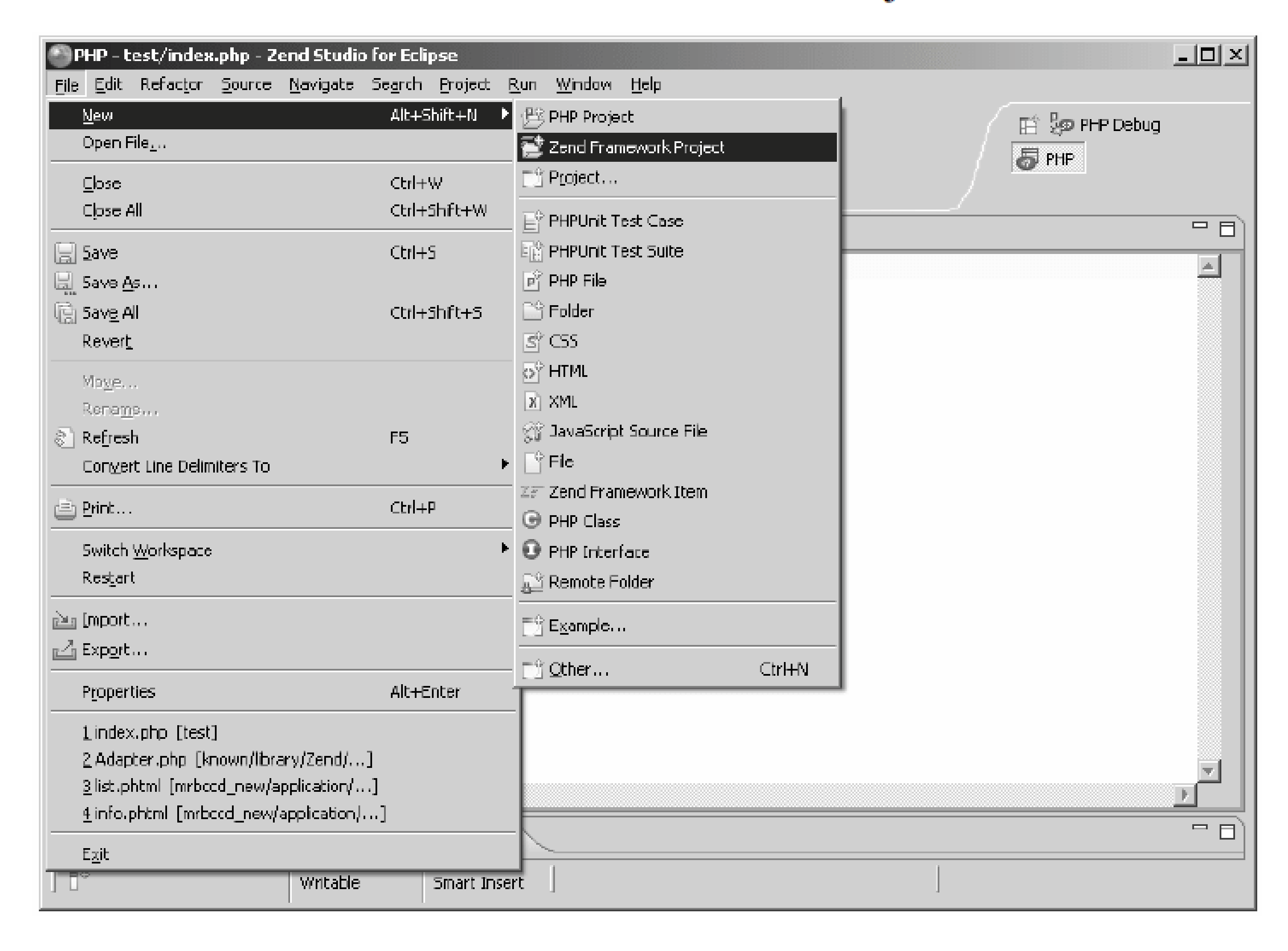


图 20.3　新建 Zend Framework 工程

（2）完成步骤（1）的操作后，将弹出如图 20.4 所示的工程设置对话框。在 Project name 文本框

中输入工程名称，这里输入“2”，在 Contents 选项区域中设置工程保存路径，既可以使用默认路径，也可以自定义新的保存路径。在 Framework Version 选项区域中选择 Zend Framework 的版本，这里选择 1.9。完成上述设置后，单击 Finish 按钮即可成功创建 Zend Framework 框架结构。

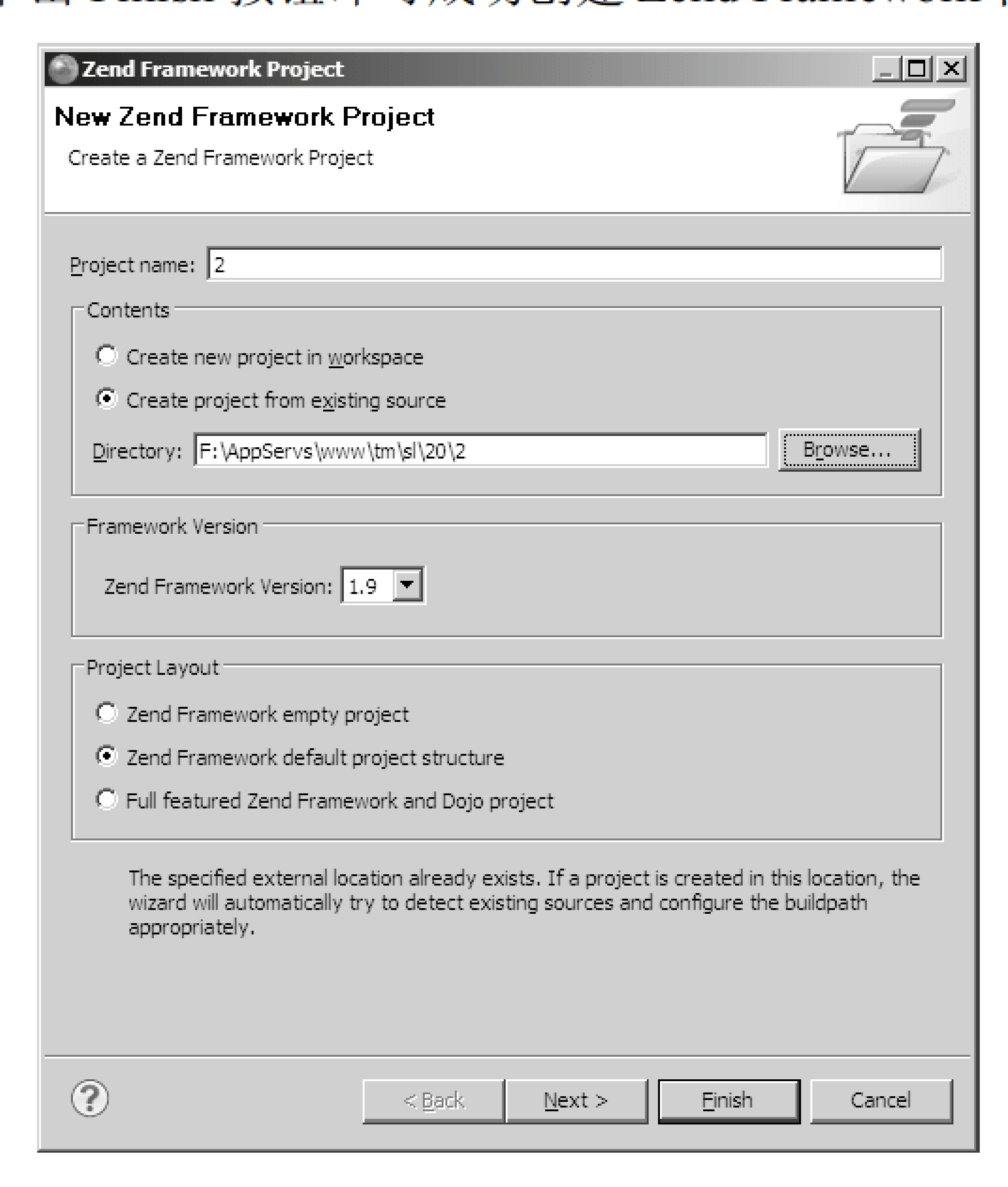


图 20.4　Zend Framework 设置对话框

（3）完成步骤（2）后即可在 Zend Studio 的左侧导航栏中预览所创建的框架结构，如图 20.5 所示。

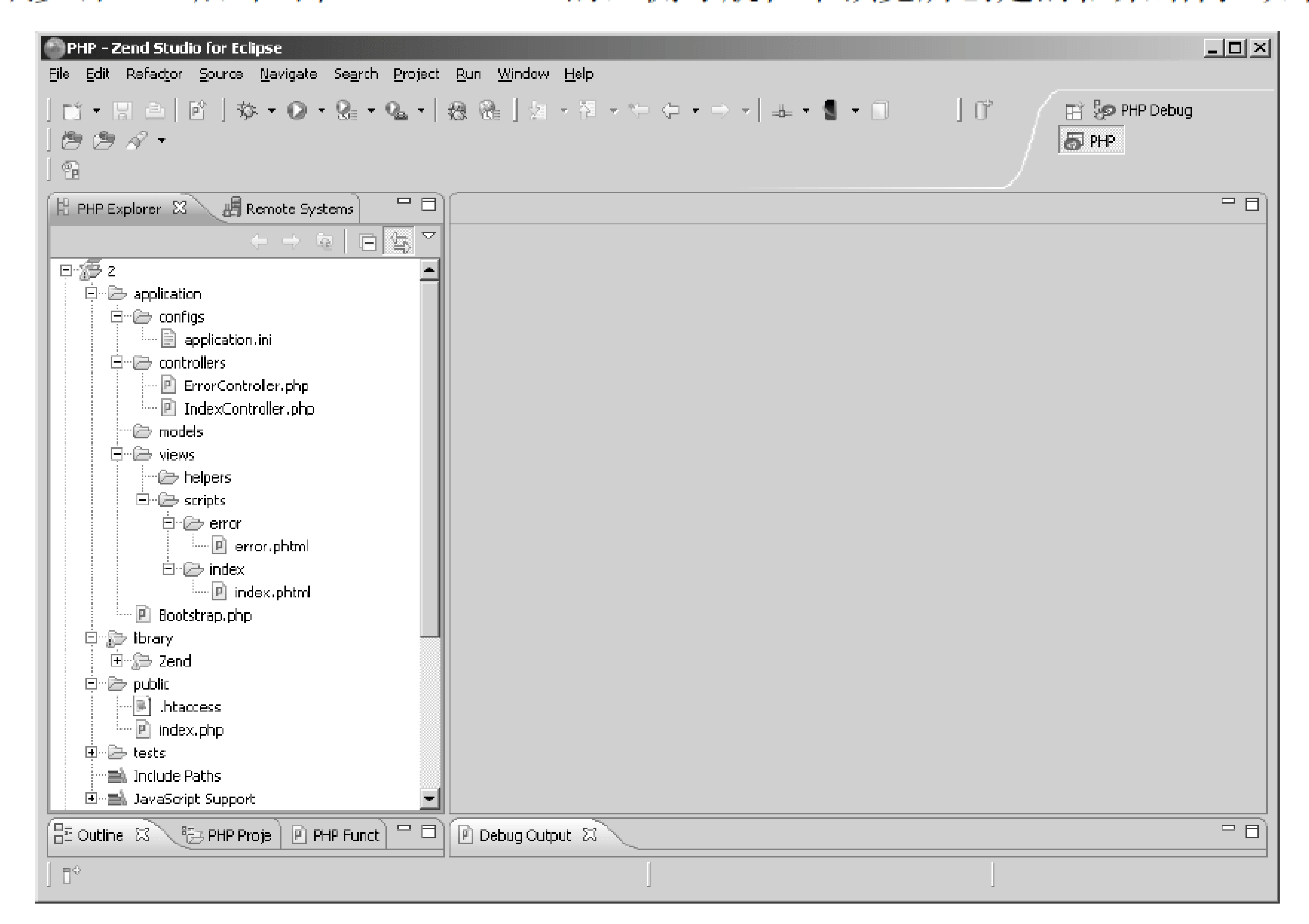
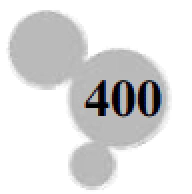


图 20.5　Zend Framework 框架结构



（4）编辑首页控制器中 indexAction 所对应的视图文件 index.phtml，在该文件中输入提示文字，然后打开浏览器，在地址栏中输入“http://localhost/tm/sl/20/2/public/”即可访问 Zend Framework 的欢迎页面，如图 20.6 所示。



图 20.6　通过 Zend Studio 成功搭建 Zend Framework 框架结构

20.3　Zend Framework 中的常用组件

Zend Framework 最大的优点就是已经将需要使用的功能封装成一个一个的类，只需要调用这些类就可以实现对应的操作。下面将介绍一些 PHP 项目中常用的 Zend Framework 组件。

20.3.1　Zend_Auth 认证

视频讲解：光盘\TM\lx\20\Zend_Auth 身份持久认证.exe

Zend_Auth 只涉及认证而不是授权。认证被宽松地定义为基于一些证书（credential）来确定一个实体（例如身份）是否确实是其所声称的。授权是一个过程，决定是否允许一个实体对其他实体进行访问、执行操作，其超出了 Zend_Auth 的范围。

注意

在 Zend_Auth 中不能使用 new 操作符和 close 关键字，而是需要使用 Zend_Auth::getInstance()来代替。

静态公共成员函数 getInstance()，实现对 Zend_Auth 的实例化，返回一个 Zend_Auth 实例，语法格式如下：

```
static Zend_Auth::getInstance ()
```

1. 适配器

Zend_Auth 适配器被用来依靠特定的认证服务（如 LDAP、RDBMS 或基于文件的存储）来认证。不同的适配器可能有不同的选项和行为，但有些基本的事情在认证适配器中是通用的。例如，接受认证证书（包括声称身份）、依靠认证服务执行查询以及返回结果在 Zend_Auth 适配器中便是通用的。

每个 Zend_Auth 适配器类都实现 Zend_Auth_Adapter_Interface。这个接口定义了一个方法 authenticate()，适配器必须为执行认证查询而实现该接口。在调用 authenticate()之前，每个适配器必须准备就绪，这样适配器准备包括设置证书（如用户名和密码）并为适配器专用的配置选项定义一些值，例如为数据库表适配器做的连接设置。

authenticate()方法用于识别不同的适配器，语法格式如下：

```
Zend_Auth::authenticate   (Zend_Auth_Adapter_Interface $adapter)
```

参数是由 Zend_Auth_Adapter_Interface 返回的适配器，返回值为 Zend_Auth_Result。

2. 结果

Zend_Auth 适配器返回一个带有 authenticate()的 Zend_Auth_Result 的实例。如果因为某些原因认证查询不能执行，authenticate()则抛出一个由 Zend_Auth_Adapter_Exception 产生的异常。

适配器基于结构组成 Zend_Auth_Result 对象，可以使用如下方法对 Zend_Auth 适配器的结果进行操作。

- ☑ isValid()：返回 true，当且仅当结果表示一个成功的认证尝试。
- ☑ getCode()：返回一个 Zend_Auth_Result 常量标识符，决定认证失败的类型或者是否认证成功。
- ☑ getIdentity()：返回认证尝试的身份。
- ☑ getMessages()：返回关于认证尝试失败的数组。

3. 身份的持久

认证一个包含认证证书的请求很有用，但是维护已认证的身份并在每次请求时不需要出示认证证书也同样很重要。

默认情况下，Zend_Auth 使用 PHP 中的 SESSION 实现身份的持久存储。基于一个成功的认证尝试，Zend_Auth::authenticate()把认证结果放入持久存储中来保存身份。

Zend_Auth 还可以使用名称为 Zend_Auth_Storage_Session 的存储类，这个类使用 Zend_Session。通过实现 Zend_Auth_Storage_Interface 给 Zend_Auth::setStorage()提供一个对象，替换一个被定制的类。

如果要使用不同的身份持久行为，则可以通过 Zend_Auth_Storage_Interface 实现，并给 Zend_Auth::setStorage()提供一个类的实例。

setStorage()方法的语法格式如下：

```
Zend_Auth::setStorage (Zend_Auth_Storage_Interface   $storage)
```

身份认证可以通过 Zend_Auth 模块，身份的持久认证则需要使用到 Zend_Session 模块，就是平时所用到的 SESSION 会话。应用 SESSION 会话只需要开启会话之后给予 SESSION 变量内容即可。下面通过两个实例了解一下 Zend_Auth 模块和 Zend_Session 模块。

【例 20.3】 会话数据页面间传递，运行结果如图 20.7 所示。实例代码如下：（**实例位置：光盘\TM\sl\20\3**）

```
require_once 'Zend/Session/Namespace.php';        //调用 Zend 框架中 Session 类
$namespace = new Zend_Session_Namespace();   //默认的命名空间
$namespace->sess = "SESSION 存在！";              //输出文字
```





图 20.7 会话数据页面间传递

注意

在项目中如果每一个页面都需要使用到 SESSION，可以在入口文件 index.php 中加入启动 SESSION；如果只是部分页面启动 SESSION，则可以在控制器中添加 SESSION 启动。

4．数据库中数据认证

在 Zend_Auth 中对数据库中数据认证使用的是 Zend_Auth_Adapter_DbTable 模块，相当于针对数据库中数据的适配器。Zend_Auth_Adapter_DbTable 的配置选项如表 20.2 所示。

表 20.2 Zend_Auth_Adapter_DbTable 配置选项

参 数	说 明
tableName	包含认证证书的数据库表名，执行数据库认证查询需要依靠这个证书
identityColumn	数据库表的列的名称，用来表示身份。身份列必须包含唯一的值
credentialColumn	数据库表的列的名称，用来表示证书。在一个简单的身份和密码认证 scheme 下，证书的值对应为密码
credentialTreatment	在许多情况下，密码和其他敏感数据是加密的。通过指定参数化的字符串来使用这个方法，如 MD5（？）或者 PASSWORD（？），开发者可以在输入证书数据时使用任意 SQL。因为这些函数对其下面的 RDBMS 是专用的。可查看数据库手册来确保所用的数据库的函数的可用性

【例 20.4】 模仿站点用户 SESSION 登录并安全退出。（**实例位置：光盘\TM\sl\20\4**）

（1）首先了解 Zend_Auth 提供的认证方式，分别为 SESSION、数据库表、摘要和 HTTP 认证适配器。本实例中使用的是 SESSION 认证，创建 Zend_Auth 认证适配器，代码如下：

```
<?php
class AuthAdapter implements Zend_Auth_Adapter_Interface
{
	protected $username;
	protected $password;
	public function __construct($username,$password)
	{
		$this->username = $username;
		$this->password = $password;
	}
	public function authenticate()
	{
		$array;
		if (($this->username == 'tm') && (($this->password == '111'))){
```

```
                $array[0] = true;
                return new Zend_Auth_Result(1,$array);
            }else{
                $array[0] = false;
                return new Zend_Auth_Result(-1,$array);
            }
        }
}
```

说明

implements 实现接口，作用是 Zend_Auth_Adapter_Interface 方法的实现。调用必须实例化实现类名称。

（2）适配器创建完成，接下来需要在控制器中调用适配器并对用户名和密码进行认证，如果用户认证成功则开启 SESSION 会话，设定 SESSION 内容。代码如下：

```
public function loginAction()
{
    $this->view->assign('title','用户登录界面');
    if ($this->_request->isPost())                                                   //是否是 POST 传输
    {
        $username = $this->_request->getPost('username');
        $password = $this->_request->getPost('password');
        $auth = Zend_Auth::getInstance();
        include '../application/class/AuthAdapter.php';
        $authAdapter = new AuthAdapter( trim($username),trim($password));      //实例化适配器
        $result = $auth->authenticate($authAdapter);                               //适配器结果
        if ($result->isValid())
        {
            $sessionNameSpace = new Zend_Session_Namespace('project');      //开启 SESSION 会话
            $sessionNameSpace->username = $username;                       //$_SESSION[username]
            $sessionNameSpace->password = $password;                       //$_SESSION[password]
            $this->_redirect('/index/success');
        }else
        {
            echo "用户名和密码错误！";
        }
    }
}
```

说明

本页面中使用的传输方式为 POST，所以需要验证是否提交需要使用$this->_request->isPost()，而接受则需要使用$this->_request->getPost()。

（3）用户身份持久认证跳转到成功界面，成功界面同样开启 SESSION，并根据登录界面中是否设定 SESSION 的值来确定是否可以查看到该页，可以防止用户直接查看页面。代码如下：



```
public function successAction()
{
    $sessionNameSpace = new Zend_Session_Namespace('project');        //开启 SESSION
    if ($sessionNameSpace->username == true)
    {
        $this->view->assign('title','权限页面');
        $this->view->assign('connect','登录成功！');
    }else
    {
        $this->_redirect('/index/login');
    }
}
```

（4）当要安全退出时只需要开启 SESSION，然后清除 SESSION，再返回登录页面即可。代码如下：

```
public function logoutAction()
{
    $sessionNameSpace = new Zend_Session_Namespace('project');        //开启 SESSION
    unset($sessionNameSpace->username);                               //删除$_SESSION['user']
    $this->_redirect('/index/login');
}
```

运行结果如图 20.8 所示。





图 20.8　模仿用户 SESSION 登录并安全退出

20.3.2　Zend_Db 数据库操作

视频讲解：光盘\TM\lx\20\Zend_Db 数据库操作.exe

Web 程序和数据库的联系是非常紧密的，所以 Zend Framework 针对数据库封装了 Zend_Db 模块，通过 Zend_Db 模块可以非常轻松地实现对数据库的操作。

在 Zend_Db 模块中，最常用的是 Zend_Db_Table 表模块。它通过 Zend_Db_Adapter 连接到数据库，以数据库模式检查表，并对该表进行操作和查询。

1．连接数据库

首先为抽象类 Zend_Db_Table 设定一个默认数据库适配器，实现与数据库的连接。连接数据库的

信息存放在 ini 配置文件中，代码如下：

```
resources.db.adapter = "PDO_MYSQL"                              //使用 PDO_MySQL 模块
resources.db.params.host = "localhost"                          //数据库连接地址
resources.db.params.username = "root"                           //数据库连接用户名
resources.db.params.password = "root"                           //数据库连接密码
resources.db.params.dbname = "db_database20"                    //数据库名称
resources.db.params.driver_options.1002 = "set names gbk"       //数据库编码方式
```



PDO_MySQL 是 Zend Framework 使用连接 MySQL 数据库的模块，这也是为什么先将 php.ini 文件中的该模块打开的原因。

2．设置表名和主键

默认情况下，Zend_Db_Table 类会将其类名当作数据库中的表名(大小写不同的地方需要添加"_")。例如一个名为 SomeTableName 的 Zend_Db_Table 类在数据库中对应表"some_table_name"，并将默认字段"id"作为表的主键。

如果不希望将类名与数据库表名以这种添加下划线的形式进行对应，并且不以"id"作为表的主键，则可以在定义表实体类时，对$_name 和$_primary 进行重构，重新定义数据表名称和 id。

在 models 文件夹下创建模型文件，定义数据表名称和 id 的关键代码如下：

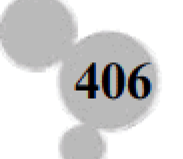
```
<?php
class Model_DbTable_User extends Zend_Db_Table
{
    protected $_name = 'tb_user';                   //数据表名
    protected $_primary = 'id';                     //主键名称
    private $_adapter;                              //设置 Zend_Db_Adapter 方法
    public function init()
    {
        $this->_adapter = $this->getAdapter();
    }
}
```

3．插入、修改和删除

（1）插入一行新数据，只需要将列名：数据的关联数组作为参数，调用 insert()方法即可。Zend Framework 会自动对数据进行加引号处理，并返回插入的最后一行的 id 值（注意：这里不同于 Zend_Db_Adapter::insert 方法，后者返回的是插入的行数）。

（2）修改表中的任意行数据，设定一个列名：数据的关联数组作为参数，调用 update()方法，同时通过一个 where 条件语句决定要修改的行。update()方法将修改表中指定行的数据，并返回被修改的行数。示例代码如下：

```
<?php
class RoundTable extends Zend_Db_Table {}
```



```
        $table = new RoundTable();
        $db = $table->getAdapter();
        $set = array(
        favorite_color' => 'yellow',                              //定义更新数据
        )
        $where = $db->quoteInto('first_name = ?', 'Robin');       //定义更新条件
        $rows_affected = $table->update($set, $where);            //执行更新操作
        //还原的更新语句：UPDATE round_table SET favorite_color = "yellow" WHERE first_name = "Robin"
?>
```

（3）删除表中的数据使用 delete()方法，并通过一个 where 条件语句指定要删除的行，该方法将返回被删除的行数。示例代码如下：

```
<?php
        class RoundTable extends Zend_Db_Table {}
        $table = new RoundTable();
        $db = $table->getAdapter();
        $where = $db->quoteInto('first_name = ?', 'Patsy');       //定义删除条件
        $rows_affected = $table->delete($where);                  //执行删除语句
        //还原的删除语句：DELETE FROM round_table WHERE first_name = "Patsy"
?>
```

4．根据主键检索、获取一条或者多条记录

（1）根据主键检索

通过调用 find()方法，可以使用主键值对表中数据进行检索。如果要查询某一条数据，则返回一个 Zend_Db_Table_Row 对象；如果要查询多条记录，则返回一个 Zend_Db_Table_Rowset 对象。示例代码如下：

```
<?php
        class RoundTable extends Zend_Db_Table {}
        $table = new RoundTable();
        // SELECT * FROM round_table WHERE id = "1"
        $row = $table->find(1);                                   //查询一条数据
        // SELECT * FROM round_table WHERE id IN("1", "2", 3")
        $rowset = $table->find(array(1, 2, 3));                   //查询多条数据
?>
```

（2）获取一条记录

通过调用 fetchRow()方法，可以完成一些非主键的查询操作。它可以附加一个 where 条件语句（和一个可选的 order 语句），返回满足条件的第一行数据的 Zend_Db_Table_row 对象。该方法与获取多条记录类似。

（3）获取多条记录

通过调用 fetchAll()方法，可以获取多条记录。它不但可以设定 where 和 order 子句，而且可以设定 limit-count 和 limit-offset 值来限制返回的结果数。该方法将选择的结果作为一个 Zend_Db_Table_ Rowset 对象返回。示例代码如下：

```
<?php
class RoundTable extends Zend_Db_Table {}
$table = new RoundTable();
$db = $table->getAdapter();
$where = $db->quoteInto('noble_title = ?', 'Sir');
$order = 'first_name';
$count = 10;
$offset = 20;
$rowset = $table->fetchAll($where, $order, $count, $offset);                    //执行查询方法
//还原语句：SELECT * FROM round_table WHERE noble_title = "Sir" ORDER BY first_name LIMIT 10 OFFSET
20
?>
```

【例 20.5】 下面通过一个具体的实例来了解一下 Zend_Db 组件中 Zend_Db_Table 模块的应用。完成对 db_database20 数据库中 tb_user 表的查询和操作。**（实例位置：光盘\TM\sl\20\5）**

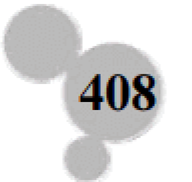
（1）这里使用在 20.2.3 节中搭建的 Zend Framework 框架，有关框架的搭建这里不再赘述。

（2）在配置文件 application.ini 中完成与数据库服务器和数据库的连接，并且设置编码格式。其关键代码如下：

```
[project]
bootstrap.path = "../application/Bootstrap.php"
bootstrap.class = "Bootstrap"
phpSettings.display_errors = 1
phpSettings.date.timezone = "Asia/GMT-8"
resources.frontController.controllerDirectory = "../application/controllers"
resources.db.adapter = "PDO_MYSQL"
resources.db.params.host = "localhost"
resources.db.params.username = "root"
resources.db.params.password = "root"
resources.db.params.dbname = "db_database20"
resources.db.params.driver_options.1002 = "set names gbk"
```

（3）直接在 models 文件夹下创建模型文件，通过$_name 和$_primary 定义数据表名称和主键。DbTable/User.php 文件的代码如下：

```
<?php
class Model_DbTable_User extends Zend_Db_Table
{
    protected $_name = 'tb_user';                    //数据表名
    protected $_primary = 'id';                      //主键名称
    private $_adapter;                               //设置 Zend_Db_Adapter 方法
    public function init()
    {
        $this->_adapter = $this->getAdapter();
    }
}
```





说明

类名称 Model_DbTable_User 根据创建的文件夹命名，如果没有创建 DbTable 文件夹，类名称为 Model_User。

（4）在控制器文件夹的 IndexController.php 中实例化此模型类，定义 indexAction()方法，使用 fetchAll()函数从模型类中读取数据；定义数据的添加方法 addAction()，通过 insert()函数完成数据的添加；定义 editAction()方法，通过 update()函数更新数据；定义 delAction()方法，通过 delete()函数删除指定的数据。关键代码如下：

```
<?php
class IndexController extends Zend_Controller_Action
{
	public function indexAction()
	{
		$users = new Model_DbTable_User();				//实例化模型类
		$this->view->assign('title','Zend_Db 数据库操作');	//定义标题变量
		$this->view->usersModel = $users->fetchAll();		//获取数据表中的数据
	}
	public function addAction()						//增加信息
	{
		$this->view->assign('title', '增加信息');
		if ($this->_request->isPost())					//判断 POST 提交数据是否为真
		{
			$validator = new Zend_Validate_NotEmpty();	//这里没有封装适配器，直接验证数据是否填写
			if ($validator->isValid($this->_request->getPost('short'))
			&& $validator->isValid($this->_request->getPost('name'))
			&& $validator->isValid($this->_request->getPost('tell'))
			&& $validator->isValid($this->_request->getPost('link'))
			&& $validator->isValid($this->_request->getPost('linktell'))
			&& $validator->isValid($this->_request->getPost('address'))
			)
			{
				$users = new Model_DbTable_User();	//实例化模型类
				$data = array(					//定义数组，添加数据
						'short' => $this->_request->getParam('short'),
						'name' => $this->_request->getParam('name'),
						'tell' => $this->_request->getParam('tell'),
						'link' => $this->_request->getParam('link'),
						'linktell' => $this->_request->getParam('linktell'),
						'address' => $this->_request->getParam('address'),
					);
				if($users->insert($data))			//添加数据
				{
					$this->_redirect('/index');		//跳转到主页
				}
			}else
			{
```

```
                echo "内容必须全部填写";                    //判断如果有项内容未填写，则输出该内容
            }
        }
    }
    public function editAction()                            //修改信息
    {
        $this->view->assign('title','修改信息');
            $id = $this->_request->getParam('id');                  //获取 GET 方式传递的 id
            if ($id > 0)
            {
                $users = new Model_DbTable_User();                  //实例化模型类
                $where = 'id=' . $id;                               //定义条件语句
                $this->view->usersModel = $users->fetchRow($where); //获取数据表指定数据
                $validator = new Zend_Validate_NotEmpty();          //实例化验证类
                if ($validator->isValid($this->_request->getParam('short'))
                && $validator->isValid($this->_request->getParam('name'))
                && $validator->isValid($this->_request->getParam('tell'))
                && $validator->isValid($this->_request->getParam('link'))
                && $validator->isValid($this->_request->getParam('linktell'))
                && $validator->isValid($this->_request->getParam('address'))
                )
                {
                    $set = array(                                   //定义更新数据数组
                        'short' => $this->_request->getParam('short'),
                        'name' => $this->_request->getParam('name'),
                        'tell' => $this->_request->getParam('tell'),
                        'link' => $this->_request->getParam('link'),
                        'linktell' => $this->_request->getParam('linktell'),
                        'address' => $this->_request->getParam('address'),
                    );
                    if ( $users->update($set, $where) )             //执行更新操作
                    {
                        $this->_redirect('/index');                 //页面跳转
                    }else
                    {
                        echo "更新失败！";
                    }
                }
            }
    }
    public function delAction()                                     //删除信息
    {
        $this->view->assign('title','删除信息');
        if ( $this->_request->isGet() )
        {
            $id = $this->_request->getParam('id');                  //获取 GET 方法提交的 id
            if ($id > 0)
            {
                $users = new Model_DbTable_User();                  //实例化模型类
```

```
                $where = 'id=' . $id;
                $users->delete($where);                         //执行删除操作
                $this->_redirect('/index');                     //页面跳转
            }
        }else
        {
            $this->_redirect('/index');                         //页面跳转
        }
    }
}
```

（5）最后，在视图文件夹 views\scripts\index 下，创建 index.phtml、add.phtml 和 edit.phtml 文件。在 index.phtml 中使用 foreach()循环输出数据表中的数据；在 add.phtml 中创建数据的添加页面；在 edit.phtml 中创建数据的修改页面。index.phtml 的关键代码如下：

```
<?php foreach($this->usersModel as $tb_user): ?>                // foreach 循环
<?php echo $this->escape($tb_user->id); ?>
<?php echo $this->escape($tb_user->short); ?>
<?php echo $this->escape($tb_user->name); ?>
<?php echo $this->escape($tb_user->tell); ?>
<?php echo $this->escape($tb_user->link); ?>
<?php echo $this->escape($tb_user->linktell); ?>
<?php echo $this->escape($tb_user->address); ?>
<?php endforeach; ?>                                            //结束 foreach 循环
```

运行结果如图 20.9 所示。

Zend_Db数据库操作　增加

客户ID	客户简称	公司名称	公司电话	联系人	联系电话	客户区域	操作	
1	IBM	International Business Machines Corp	010-88888888	jeck	13888888888	USA	修改	删除
2	Microsoft	Microsoft	010-66666666	Bill	13666666666	USA	修改	删除
31	1	1	1	1	1	1	修改	删除

图 20.9　数据库数据显示

20.3.3　Zend_Cache 精细缓存

视频讲解：光盘\TM\lx\20\Zend_Cache 精细缓存.exe

对于大型的企业级网站，数据量和并发数量都非常的大，如果单纯地读取数据库将使站点速度和效率降低，这时最好的解决办法是将数据或页面存储到内存中，这样就不用每一次查看数据时都从数据库中读取，从而使网站运行速度大大提高。

一般应用缓存有如下 3 点要求：

☑ 生成页面时，数据输出是相同的，如首页。

☑ 如果存在敏感信息的输入，则产生的页面不可以使用缓存，如注册页面。

☑ 信息更新非常频繁的数据和页面不可以使用缓存，如聊天室。

合理地应用缓存可以使站点的运行速度大大提升，也可以减轻对数据库的操作次数。

Zend Framework 中通过 Zend_Cache 缓存数据，Zend_Cache 不但使用方便，而且可以对数据或者某一文件夹下的页面进行缓存。

Zend Framework 将 Zend_Cache 缓存分为前端和后端，前端用于操作缓存，后端提供缓存的存储方式。可以通过设置前端和后端的参数控制缓存。

前端选项中参数值如表 20.3 所示。

表 20.3　Zend_Cache 前端选项参数值

参　　数	类　　型	默　认　值	说　　明
caching	Bool	true	打开/关闭缓存（对被缓存脚本的调试非常有用）
cache_id_prefix	String	null	所有缓存 id 的前缀，如果设置为 null，没有缓存 id 前缀使用。缓存 id 前缀在缓存中创建一个命名空间，允许多个程序和网站共享缓存。每个程序或网站可以使用不同的缓存 id 前缀，所以特定的缓存 id 可以使用多次
lifetime	Int	3600	缓存生命期（秒），如果设置为 null，缓存永远有效
logging	Bool	false	如果设置为 true，日志记录（通过使用 Zend_Log）被激活（但是系统将变慢）
write_control	Bool	true	打开/关闭写控制（the cache is read just after writing to detect corrupt entries），打开写控制轻微地放慢缓存写的速度但不影响读（it can detect some corrupt cache files but it's not a perfect control）
automatic_serialization	Bool	false	打开/关闭自动序列化，可以直接用于保存非字符串数据（但是很慢）
automatic_cleaning_factor	Int	10	关闭/调整自动清理过程（垃圾收集器）：0 表示不自动清理缓存，1 表示自动清理缓存，并且如果 x>1，表示 x 写操作后自动随机清理 1 次
ignore_user_abort	Bool	false	如果设置为 true，核心将在 save()方法中设置 ignore_user_abort PHP flag，以免在某些情况下缓存崩溃

后端选项中参数值如表 20.4 所示。

表 20.4　Zend_Cache 后端选项参数值

参　　数	类　　型	默　认　值	说　　明
cache_dir	String	/tmp/	存放缓存文件的目录
file_locking	Bool	true	启用/禁用文件锁定：在比较坏的情况下，能够避免缓存中断，但在多线程 Web 服务器或者 NFS 文件系统中没有任何帮助
read_control	Bool	true	启用/禁用读控制：如果启用，控制键被嵌入到缓存文件中，并且这个键将与读取后计算出的值进行比较
read_control_type	String	crc32	读控制类型（仅在读控制启用时）。可用的值有: 'md5'（最好但最慢），'crc32'（安全性稍差，但更快，更好的选择），'adler32'（新选择，比 crc32 快），'strlen' for a length only test（最快）

续表

参 数	类 型	默 认 值	说 明
hashed_directory_level	Int	0	Hash 目录结构层次：0 表示“不使用 hash 的目录结构”，1 表示“一级目录结构”，2 表示“二级目录结构”……次选项在有非常多的缓存文件时能够加速缓存。只有相关的基准测试才能帮助用户选择合适的值。也许 1 或 2 是一个好的开始
hashed_directory_umask	Int	0700	目录结构的 UNIX 掩码
file_name_prefix	String	zend_cache	缓存文件前缀。小心使用此选项，因为当清除缓存时，在系统缓存目录（像/tmp）中一个太 generic 的值将导致灾难
cache_file_umask	Int	0700	缓存文件掩码
metatadatas_array_max_size	Int	100	metadatas 数组的内部最大尺寸（不要随意更改这个值）

下面通过具体实例来了解一下 Zend_Cache 缓存的应用。

【例 20.6】 缓存数据库中读取出的数据，并设置缓存时间为 1 分钟。程序第一次在本机上运行时，是从数据库中读取数据，但是在执行一次刷新或者重新在本机中运行程序时，则是从缓存中读取数据。**（实例位置：光盘\TM\sl\20\6）**

（1）这里在实例 20.4 的基础上，应用 Zend_Cache 缓存数据。缓存的设置在启动文件 Bootstrap.php 的_init Cache 方法中完成，代码如下：

```
public function _initCache()
{
    $frontendOptions = array ('lifeTime' => 60,                        //设置缓存时间为 60，即 1 分钟
                              'automatic_serialization' => true );
    $backendOptions = array ('cache_dir' => '../application/tmp/' );
    $cache = Zend_Cache::factory ( 'Core', 'File', $frontendOptions, $backendOptions );
    Zend_Registry::set ( 'cache', $cache );                            //设置名称
}
```

说明

Zend_Cache::factory()为 Zend_Cache 的工厂方法，4 个参数分别为前端名称、后端名称、前端选项和后端选项。

（2）在模型文件中，将从数据库中读取的数据放入名称为 userinfo 的缓存中，并根据缓存名称返回结果集，代码如下：

```
public function findAll()
{
    $cache = Zend_Registry::get ( 'cache' );
    if (! $result = $cache->load ( 'userinfo' )) {                    //是否存在名称为 userinfo 的缓存
        $result = $this->fetchAll();                                 //使用 fetchAll()函数将数据读取
        $cache->save ( $result, 'userinfo' );                        //将数据存入 userinfo 缓存中
```

```
    }
    return $result;
}
```

（3）为了使页面中能够区分是从数据库中读取数据，还是从缓存中读取数据，这里对读取数据结果作一个 if 判断，代码如下：

```
$cache = Zend_Registry::get ( 'cache' );                    //工厂模式开始 Zend_Cache 缓存
if (! $result = $cache->load ( 'userinfo' ))                //是否存在 userinfo 名称的缓存
{
    $this->view->content = "从数据库中读取数据";
    $this->view->usersModel = $users->findAll();
}else
{
    $this->view->content = "从内存中读取数据,时间设置为 1 分钟";
    $this->view->usersModel = $users->findAll();
}
```

运行本实例，第一次将输出如图 20.10 所示的页面；当刷新页面或者重新运行程序时，则将输出如图 20.11 所示的页面。

Zend_Db数据库操作　从数据库中读取数据

ID	客户	公司名称	公司电话	联系人	联系电话	客户区域
1	IBM	International Business Machines Corp	010-88888888	jeck	13888888888	USA
2	Microsoft	Microsoft	010-66666666	Bill	13666666666	USA

图 20.10　从数据库中读取数据

Zend_Db数据库操作　从内存中读取数据,时间设置为1分钟

ID	客户	公司名称	公司电话	联系人	联系电话	客户区域
1	IBM	International Business Machines Corp	010-88888888	jeck	13888888888	USA
2	Microsoft	Microsoft	010-66666666	Bill	13666666666	USA

图 20.11　从内存中读取数据

20.3.4　Zend_Layout 站点布局

视频讲解：光盘\TM\lx\20\Zend_Layout 站点布局.exe

在网站中很多时候头部文件和尾部文件是不变的，这时如果对每一个页面都加入头部文件和尾部文件是一件非常麻烦的事情，并且修改起来也非常繁琐。

Zend Framework 提供了一个简单而方便的布局模块 Zend_Layout，它将相同的头部和尾部代码存储在独立的布局显示脚本（layout view script）中，并在布局显示脚本中包含与正在执行的动作相关的显示代码。



要应用 Zend_Layout，必须对 Zend Framework 框架进行以下修改：

（1）确定布局显示脚本的存储位置。建议存储在 application 目录下，在这个目录下创建 layouts 文件夹，用于存储布局显示脚本。

（2）通知启动文件启用 Zend_Layout。在 application/configs/application.ini 文件中添加如下代码：

```
resources.layout.layoutPath="../application/layouts"
```

（3）创建布局显示脚本。布局显示脚本默认名称是 layout.phtml，存储在 layouts 文件夹下。在这个布局显示脚本中可以使用：

☑ escape()函数，确保标题内容被恰当地编码。

☑ layout()辅助函数，将内容在 content 占位符中输出，确保动作显示脚本在布局显示脚本之前执行。

（4）创建布局显示脚本调用的动作显示脚本的内容。

上述就是应用 Zend_Layout 布局网站的流程。下面通过一个简单的 Zend_Layout 实例了解一下它的应用。

【例 20.7】 使用 Zend_Layout 对网站进行布局，将网页中头文件和尾文件分离。**（实例位置：光盘\TM\sl\20\7）**

（1）以 20.2.3 节中创建的 Zend Framework 框架为基础，在应用程序文件夹中创建 layouts 文件夹。

（2）通知启动文件启用 Zend_Layout。在 application/configs/application.ini 文件中添加如下代码：

```
resources.layout.layoutPath="../application/layouts"
```

（3）在 layouts 文件夹中创建布局显示脚本 default.phtml（名称自定义），完成页面布局。关键代码如下：

```
<?php echo $this->partial('header.phtml'); ?>
<?php echo $this->partial('index/index.phtml'); ?>
<?php echo $this->partial('footer.phtml'); ?>
```

（4）在视图 views/scripts 文件夹下创建 header.phtml（头文件）和 footer.phtml（尾文件）。

（5）在控制器文件 IndexController.php 中，设置被访问的布局。加入代码如下：

```
public function init()
{
	$this->_helper->layout->setLayout ( "default" );
	$this->view->assign ( "title", "Zend_Layout 网站布局" );
}
```


说明

$this->_helper->layout->setLayout ("default")中的 default 是设定的 layout 中的文件名称。

运行结果如图 20.12 所示。



图 20.12　Zend_Layout 站点布局

20.3.5　Zend_Paginator 数据分页

视频讲解：光盘\TM\lx\20\Zend_Paginator 数据分页.exe

如果一个页面中数据过多会使运行速度降低，并且占用非常大的内存，并有可能造成站点崩溃。处理此问题最好的方法就是将数据进行分页显示，即不占用过多的内存，而且可以提供网页运行速度。

Zend Framework 中提供了 Zend_Paginator 模块，该模块可以对数据集分页输出。其主要特点如下：

☑　对任意数据进行分页，而非专门针对关系型数据库。

☑　只取出需要用来进行呈现的数据（有用数据）。

☑　不强制用户只使用一种途径呈现数据和渲染分页控件。

☑　可以单独使用（松藕性）。

如果想要非常漂亮或者非常有个性的页码模板视图，可以通过修改页码模板中的占位符来实现。常用占位符如表 20.5 所示。

表 20.5　页码视图模板占位符说明

参　　数	说　　明
first	第一页的页码
firstItemNumber	当前页的第一条记录在整个记录集中的位置
firstPageInRange	第一次运行现实的页码
current	当前页码
currentItemCount	当前页记录数
itemCountPerPage	本页最多记录数



续表

参　　数	说　　明
last	最后页码
lastItemNumber	当前页的最后一条记录在整个记录集中的位置
next	下一页页码
pageCount	总页数
pagesInRange	现实页码数组
previous	上一页页码
totalItemCount	总记录数

下面通过详细讲解一个实例来理解 Zend_Paginator 数据分页。

【例 20.8】 使用 Zend_Paginator 对数据表 tb_user 实现分页显示。**（实例位置：光盘\TM\sl\20\8）**

根据实例 20.4 从数据库中读取数据程序，只需要增加和修改以下文件。

（1）通过对页码占位符的学习，现在可以编辑页码分页，在视图文件夹的 index 下创建一个 pagelist.phtml 文件（文件名随便起），代码如下：

```
<?php if ($this->pageCount): ?>
<div class="paginationControl">
<!--上一页连接 -->
<?php if (isset($this->previous)): ?>
   <a href="<?php echo $this->url(array('controller'=>'index', 'action'=>'index', 'page'=>$this->previous)); ?>"><
上一页</a> |
<?php else: ?>
   <span>< 上一页</span> |
<?php endif; ?>
<!--数字连接 -->
<?php foreach ($this->pagesInRange as $page): ?>
  <?php if ($page != $this->current): ?>
     <a href="<?php echo $this->url(array('controller'=>'index', 'action'=>'index', 'page'=>$page)); ?>"><?=
$page; ?></a> |
  <?php else: ?>
    <strong><?= $page; ?></strong> |
  <?php endif; ?>
<?php endforeach; ?>
<!--下一页连接 -->
<?php if (isset($this->next)): ?>
  <a href="<?php echo $this->url(array('controller'=>'index', 'action'=>'index', 'page'=>$this->next)); ?>">下一页 >
</a>
<?php else: ?>
   <span>下一页  ></span>
<?php endif; ?>
</div>
<?php endif; ?>
```

（2）在 index.phtml 文件中必须存在调用此分页文件，代码如下：

```
<?php echo $this->paginationControl($this->paginator, 'Elastic', 'index/pagelist.phtml'); ?>
```



调用 paginationControl()函数，给出 3 个参数，第一个参数是将控制器中的 paginator 调用，第二个参数为调用分页样式，第 3 个参数为页码分页所在地址。第二个参数的详细分类如表 20.6 所示。

表 20.6　样式详细分类说明

参　数	说　明
All	返回每个页码，这是下拉菜单中的分页，适合页面较少的分页
Elastic	类似于 Google 的分页样式
Jumping	跳转到最后页码之后，就重新开始页码
Sliding	类似于雅虎分页样式，以当前页为中心

（3）分页调用之后需要对数据库中的数据重新设计，下面开始对控制器加入读取数据样式和分页样式，代码如下：

```
/* 开始分页 */
pageNumber = 3;                                                    //每页数据数量 常量
$paginator = Zend_Paginator::factory($this->view->usersModel);    //使用 Zend_Paginator 工厂方式
$paginator->setItemCountPerPage($pageNumber);                     //将每页数据数量带入
$paginator->setCurrentPageNumber($this->_getParam('page'));       //获取当前页数
Zend_Paginator::setDefaultScrollingStyle('Sliding');              //设置样式
$paginator->setView($this->view);                                 //视图数据输出
$this->view->usersModel = $paginator;
$this->view->paginator = $paginator;
$this->render();
```

说明

首先是使用 Zend_Paginator 的工厂方法将数据表中的数据得到，然后设定每一页显示的数据数量和使用 GET 方式得到页码数，并设置样式。样式的设置已在表 20.6 中详细讲解。这时开始将视图数据输出到页面中。

运行结果如图 20.13 所示。

Zend_Db数据库操作

客户ID	客户简称	公司名称	公司电话	联系人	联系电话	客户区域
1	IBM	International Business Machines Corp	010-88888888	jeck	13888888888	USA
2	Microsoft	Microsoft	010-66666666	Bill	13666666666	USA
31	明日科技	明日科技有限公司	0431-88888888	刘女士	13866886688	长春市

< 上一页 | 1 | 2 | 下一页 >

图 20.13　Zend_Paginator 数据分页



【例 20.9】 实例 20.8 是对正常分页模式的讲解，下面对下拉菜单分页进行详细讲解。**（实例位置：光盘\TM\sl\20\9）**

（1）如果需要下拉菜单分页，首先需要设置跳转页面 js 函数，然后需要修改 pagelist.phtml 文件。在程序运行根目录 public 文件夹中加入 js/jump.js 文件，代码如下：

```
function MM_jumpMenu(targ,selObj,restore){ //v3.0
  eval(targ+".location='"+selObj.options[selObj.selectedIndex].value+"'");
  if (restore) selObj.selectedIndex=0;
}
```


说明

这个函数是跳转页面 js 函数，在 Dreamweaver 中可以直接得到。

（2）在页面中调用此 js 函数，在 index.phtml 文件中加入如下代码：

```
<script type="text/javascript" src="<?php echo $this->baseUrl("js/jump.js") ?>"></script>
```

（3）更改 pagelist.phtml 文件，实现跳转菜单分页，代码如下：

```
<?php if ($this->pageCount): ?>
<div class="paginationControl">
<!--上一页连接 -->
<?php if (isset($this->previous)): ?>
    <a href="<?php echo $this->url(array('controller'=>'index', 'action'=>'index', 'page'=>$this->previous)); ?>">< 
上一页</a>
<?php else: ?>
    <span>< 上一页</span>
<?php endif; ?>
<!--下拉菜单连接 -->
<select name="menu1" onchange="MM_jumpMenu('parent',this,0)">
<?php foreach ($this->pagesInRange as $page): ?>
<?php if ($page != $this->current): ?>
      <option value="<?php echo $this->url(array('controller'=>'index', 'action'=>'index', 'page'=>$page)); ?>"><?=
$page; ?></option>
<?php else: ?>
      <option selected="selected"><?= $this->current ?></option>
<?php endif; ?>
<?php endforeach; ?>
</select>
<!--下一页连接 -->
<?php if (isset($this->next)): ?>
    <a href="<?php echo $this->url(array('controller'=>'index', 'action'=>'index', 'page'=>$this->next)); ?>">下一
页 ></a>
<?php else: ?>
```

```
    <span>下一页　></span>
<?php endif; ?>
</div>
<?php endif; ?>
```


说明

简单下拉菜单分页不适合于数据较多的情况。

运行结果如图 20.14 所示。

Zend_Db数据库操作

客户ID	客户简称	公司名称	公司电话	联系人	联系电话	客户区域
1	IBM	International Business Machines Corp	010-88888888	jeck	13888888888	USA
2	Microsoft	Microsoft	010-66666666	Bill	13666666666	USA
31	明日科技	明日科技有限公司	0431-88888888	刘女士	13866886688	长春市

< 上一页 1 下一页 >

图 20.14　Zend_Paginator 下拉菜单分页

20.3.6　Zend_Mail 发送邮件

视频讲解：光盘\TM\lx\20\Zend_Mail 发送邮件.exe

Zend_Mail 提供了通用化的功能来撰写和发送文本以及兼容 MIME 标准的含有多个段的邮件消息。Zend_Mail 支持默认的 Zend_Mail_Transport_Sendmail 传输或 Zend_Mail_Transport_Smtp 发送。

Zend_Mail 将通过 mail()函数和 SMTP 实现发送邮件、接收邮件、发送附件、接收附件等功能，并且都封装到类中。通过对类中方法的调用，即可实现相应的功能。

这里只讲解通过 Zend_Mail 中的 SMTP 发送邮件的方法，并且支持 SMTP 身份验证。Zend_Mail 发送邮件应用到的函数如表 20.7 所示。

表 20.7　Zend_Mail 发送邮件应用的基本函数

函　　数	说　　明
addTo($option,$to)	收件人地址。$option 为收件人地址，$to 为显示收件人地址
setFrom($option,$to)	发件人地址。$option 为发件人地址，$to 为显示发件人地址
setSubject($subject)	邮件标题
setbodyText($body)	文本格式邮件内容
setbodyHtml($body)	HTML 格式邮件内容
send($transport)	执行发送，参数是实例化验证类返回的对象

在 Zend_Mail_Transport_Smtp 中完成身份验证，使用 Zend_Mail 发送 Html 邮件，示例代码如下：

```
<?php
    require_once 'Zend/Mail.php';
```



```
    require_once 'Zend/Mail/Transport/Smtp.php';
    $config = array('auth' => 'login',
              'username' => 'mr',
              'password' => 'mrsoft');                            //定义 SMTP 的验证参数
    $transport = new Zend_Mail_Transport_Smtp('smtp.sohu.com', $config);  //实例化验证的对象
    $mail = new Zend_Mail('GBK');                                 //实例化发送邮件对象
    $mail->setBodyHtml($mailbody);                                //发送邮件主体
    $mail->setFrom($envelope, '明日科技编程词典用户注册');          //定义邮件发送使用的邮箱
    $mail->addTo($_GET[email], '获取用户注册激活码');               //定义邮件的接收邮箱
    $mail->setSubject('获取注册用户的激活码');                      //定义邮件主题
    $mail->send($transport);                                      //执行发送操作
```

【例 20.10】 本实例应用 Zend_Mail 模块实现向互联网中指定的邮箱中发送邮件。（**实例位置：光盘\TM\sl\20\10**）

程序的开发步骤如下：

（1）在程序根目录文件 public 下建立程序入口文件 index.php，写入代码如下：

```
<?php
set_include_path('../library'.PATH_SEPARATOR.get_include_path());                 //设定路径
require_once 'Zend/Application.php';                                              //调用 Zend 类库
$application = new Zend_Application('project','../application/configs/application.ini');  //读取配置文件
$application->bootstrap()->run();                                                 //运行方式　启动文件->运行
```



说明

入口文件功能为设定程序路径，并调用包含类 Zend_Application 将 Zend Framework 中类全部带入，使用的方式为 oop 工厂模式。

（2）在程序根目录文件 public 下建立 URL 重载文件.htaccess，写入代码如下：

```
RewriteEngine on
RewriteRule !\.(js|css|gif|jpg|png)$ index.php
```



说明

URL 重载代码意思为如果文件后缀名称为 js|css|gif|jpg|png 格式，跳转到入口文件 index.php。

（3）建立应用程序文件夹 application，根据入口文件 index.php 写入配置文件信息来建立 configs/application.ini 文件，写入代码如下：

```
[project]
bootstrap.path = "../application/Bootstrap.php"                          //设置启动文件路径
bootstrap.class = "Bootstrap"                                            //启动类名称
phpSettings.display_errors = 1                                           //错误输出
phpSettings.date.timezone = "Asia/GMT-8"                                 //时间区域
resources.frontController.controllerDirectory = "../application/controllers"  //控制器文件路径
```

说明

配置文件中的启动类名称默认为 Bootstrap，可以通过该设置设定另外的名称。如果不给出时间区域则时间会完全按照格林威治时间显示。

（4）在应用程序文件夹 application 下建立启动文件 Bootstrap.php 文件，写入代码如下：

```
<?php
class Bootstrap extends Zend_Application_Bootstrap_Bootstrap
{
	public function _initAutoload()
	{
		$moduleAutoloader = new Zend_Application_Module_Autoloader(array('namespace' => '','basePath'
=> '../application'));
		return $moduleAutoloader;
	}
}
```

说明

启动文件中以_init 为开头的函数被 Zend Framework 视为自动加载函数。如果需要在启动文件中加入另外的函数，必须以_init 为开头。

（5）在控制器文件夹中建立 IndexController.php 文件，调用 Zend_Mail_Transport_Smtp()类对登录邮箱进行验证，调用 Zend_Mail 类发送邮件。IndexController.php 的代码如下：

```
public function indexAction()
{
	$this->view->assign('title','Zend_Mail 发送邮件');
	$validator = new Zend_Validate_NotEmpty();
	$config = array('auth' => 'login',
			'username' => 'cym3100',
			'password' => '851112');				//定义 SMTP 的验证参数，验证登录邮箱
	if ($this->_request->isPost())
	{
		if ( $validator->isValid($this->_request->getPost("to"))			//验证收件人是否为空
		&&	$validator->isValid($this->_request->getPost("from"))		//验证发件人是否为空
		&& $validator->isValid($this->_request->getPost("subject"))		//验证主题是否为空
		&& $validator->isValid($this->_request->getPost("body"))		//验证内容是否为空
		)
		{
			$tr = new Zend_Mail_Transport_Smtp('mail.sohu.com', $config);	//调用外部邮箱信息
			$mail = new Zend_Mail();
			$mail->addTo($this->_request->getPost('to'),$this->_request->getPost('to'));		//收件人信息
			$mail->setFrom($this->_request->getPost('from'),$this->_request->getPost('from'));//发件人信息
			$mail->setSubject($this->_request->getPost('subject'));				//主题信息
			$mail->setBodyText($this->_request->getPost('body'));				//内容信息
```

```
                $mail->send($tr);                                    //邮件发送
                $this->_redirect('/');                               //返回本页
            }else
            {
                echo "全部信息必须填写";
            }
        }
}
```

说明

测试本实例，需要连接到互联网。这里通过 Zend_Mail_Transport_Smtp 类登录到互联网中的搜狐邮箱（邮箱：cym3100@souhu.com，密码：851112），然后以这个邮箱作为发件人，向其他邮箱中发送邮件。这里的发件人，读者可以根据自己的需要进行修改。

注意

邮件发送成功并不代表邮件已经到达邮箱，所以测试时需要注意。

（6）最后在视图文件 views 中的 scripts/index 下创建文件 index.phtml，其关键代码如下：

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title><?php echo $this->escape($this->title); ?></title>
<link href="<?php echo $this->baseUrl("css/css.css"); ?>" rel="stylesheet" type="text/css" />
</head>
<body>
<table width="743" height="550" border="0" align="center" cellpadding="0" cellspacing="0" background="<?php
echo $this->baseUrl("images/mail_02.jpg"); ?>">
  <tr>
    <td valign="top">
     <form id="form1" name="form1" method="post" action="">
     <table width="100%" border="0" cellspacing="0" cellpadding="0">
       <tr>
         <td align="right" class="black_12_28">收信人：</td>
         <td><input name="to" type="text" size="70" /></td>
       </tr>
       <tr>
         <td align="right" class="black_12_28">发件人：</td>
         <td><input name="from" type="text" size="70" /></td>
       </tr>
       <tr>
         <td align="right" class="black_12_28">主题：</td>
         <td><input name="subject" type="text" size="70" /></td>
       </tr>
       <tr>
```

```
            <td align="right" valign="top" class="black_12_28">内容：</td>
            <td align="left" valign="top"><textarea name="body" cols="68" rows="12"></textarea></td>
          </tr>
          <tr>
            <td align="center" valign="top">
            <input  type="submit"  name="Submit"  value="  发  送  "  style="background:url(<?php  echo
$this->baseUrl ("images/submit.jpg"); ?>);width:75px;height:23px;border:0px;"/>
              </td>
          </tr>
        </table>
         </form>
         </td>
     </tr>
</table>
</body>
</html>
```



使用 form 表单时需要注意的是 action 跳转都是不加入任何页面的，因为处理都是在控制层中。

运行结果如图 20.15 所示。

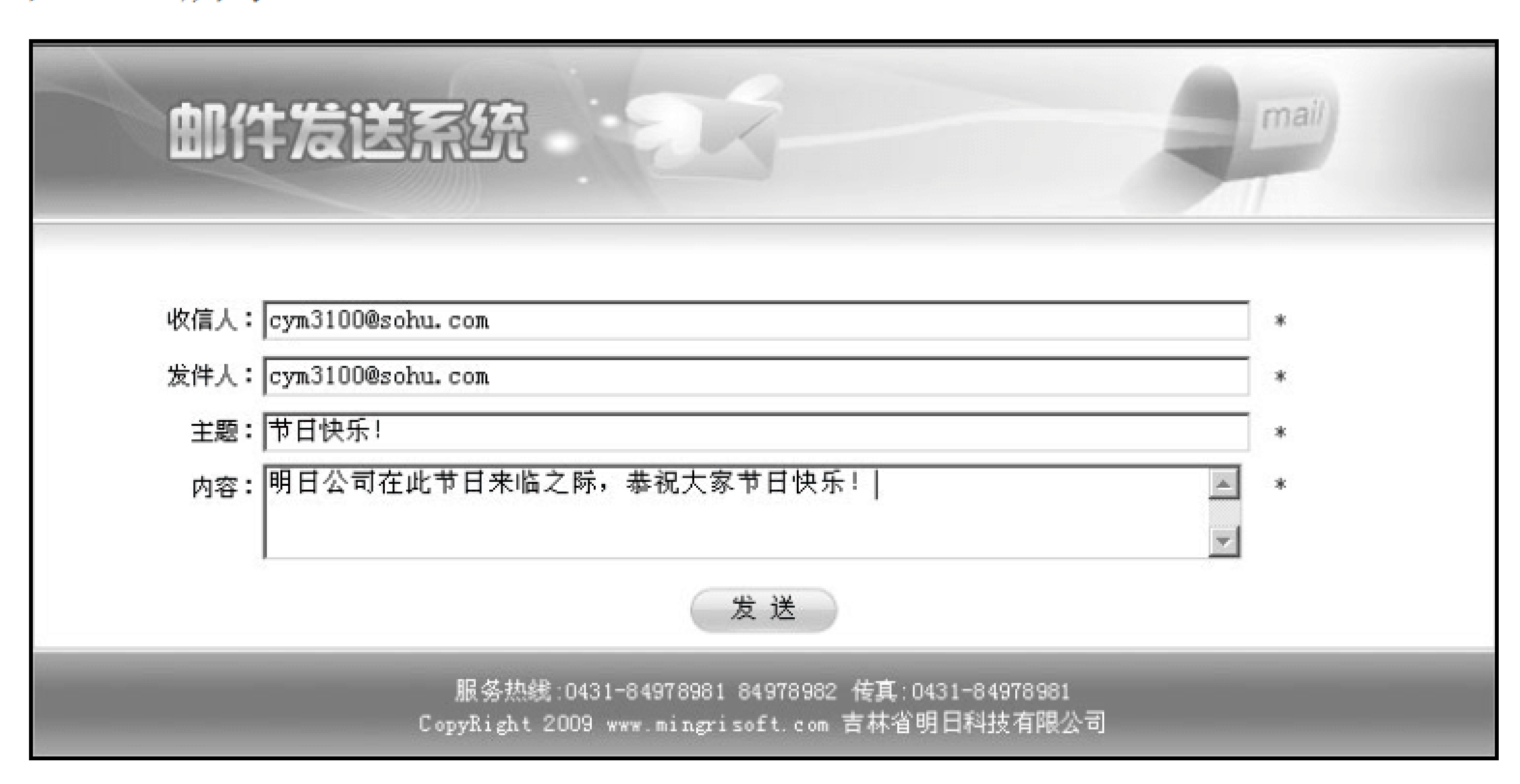


图 20.15　Zend_Mail 发送邮件

20.4　小　　结

本章讲述了 Zend Framework 搭建环境的要求和步骤，对 Zend Framework 的常用功能进行了详细的讲解，最后通过 Zend Framework 开发一个邮件发送模块，熟悉 Zend Framework 开发 Web 程序的步骤，同时在练习与实践中开发了一个留言板模块。相信通过本章的学习，读者对 Zend Framework 能够有一定的了解和认识，能够为读者以后应用 Zend Framework 做好铺垫。



20.5　练习与实践

使用 Zend Framework 框架制作一个留言板，运行结果如图 20.16 所示。（**答案位置：光盘\TM\sl\20\11**）



图 20.16　留言板

第 3 篇

高级应用

本篇介绍了 Smarty 模板技术、PHP 与 XML 技术和 PHP 与 Ajax 技术等。掌握本篇内容后，能够开发一些实用的网络程序等。

第21章

Smarty 模板技术

（视频讲解：55 分钟）

目前网络上针对 PHP 的模板数不胜数。作为最早的 MVC 模板之一，Smarty 在功能和速度上处于绝对的领先优势。那么，Smarty 的特点是什么？它是如何完成代码分离的呢？

通过阅读本章，您可以：

- 了解什么是 MVC，什么是 Smarty
- 掌握 Smarty 的特点
- 掌握 Smarty 模板的安装和配置的方法
- 掌握 Smarty 模板设计的方法
- 掌握 Smarty 程序设计的方法
- 熟悉 Smarty 模板的应用

21.1　Smarty 简介

视频讲解：光盘\TM\lx\21\Smarty 简介.exe

Smarty 是 PHP 中的一个模板引擎，是众多 PHP 模板中最优秀、最著名的模板之一。

21.1.1　什么是 Smarty

Smarty 是一个使用 PHP 编写的 PHP 模板引擎，它将一个应用程序分成两部分实现：视图和逻辑控制。简单地讲，目的就是将 UI（用户界面）和 PHP code（PHP 代码）分离。这样，程序员在修改程序时不会影响到页面设计，而美工在重新设计或修改页面时也不会影响程序逻辑。

21.1.2　Smarty 与 MVC

Smarty 这种开发模式，正是基于 MVC 框架概念。

MVC（Model-View-Controller），即模型-视图-控制器，是指一个应用程序由 3 部分构成：模型部分、视图部分和控制部分。

- ☑ 模型：对接收的信息进行处理，并将处理结果回传给视图。例如，如果用户输入信息正确，那么将给视图一个命令，允许用户进入主页面，反之则拒绝用户的操作。
- ☑ 视图：就是提供给用户的界面。视图只提供信息的收集及显示，不涉及处理。如用户登录界面，也就是视图，只提供用户登录的用户名和密码输入框（也可以有验证码、安全问题等信息），至于用户名和密码的对与错，这里不去处理，直接传给后面的控制部分。
- ☑ 控制：负责处理视图和模型的对应关系，并将视图收集的信息传递给对应的模型。例如，当用户输入用户名和密码后提交，这时，控制部分接收用户的提交信息，并判断这是一个登录操作，随后将提交信息转发给登录模块部分，也就是模型。

21.1.3　Smarty 特点

- ☑ 采用 Smarty 模板编写的程序可以获得最快的速度。注意，这是相对于其他模板而言。
- ☑ 可以自行设置模板定界符，如{}、{{}}、<!--{}-->等。
- ☑ 仅对修改过的模板文件进行重新编译。
- ☑ 模板中可以使用 if/elseif/else/endif。
- ☑ 内建缓存支持。
- ☑ 可自定义插件。



21.2　Smarty 的安装配置

视频讲解：光盘\TM\lx\21\Smarty 模板的安装配置.exe

21.2.1　Smarty 下载和安装

PHP 没有内置 Smarty 模板类，需要单独下载和配置，而且 Smarty 要求服务器上的 PHP 版本最低为 4.0.6。用户可以通过访问 http://smarty.php.,net/download.php 下载最新的 Smarty 压缩包。本章使用的版本是 Smarty-2.6.19。

将压缩包解压后，得到一个 libs 目录，其中包含了 Smarty 类库的核心文件，包括 smarty.class.php、smarty_Compiler.class.php、config_File.class.php 和 debug.html 4 个文件，另外还有 internals 和 plug-ins 两个目录。复制 libs 目录到服务器根目录下，并为其重命名，一般该目录的名称为 smarty 或 class 等，这里改为 smarty。至此，Smarty 模板安装完毕。

注意

凡是在后面的章节中提到 Smarty 类包、Smarty 目录等，都是这个重命名后的 Smarty，即原 libs 目录。

21.2.2　第一个 Smarty 程序

使用 Smarty 模板不像 Smarty 手册或有些书籍中讲的那么复杂、繁琐。这里先实现第一个 Smarty 实例，并对过程进行讲解。对 Smarty 有了初步了解后，再学习 Smarty 的配置信息。

【例 21.1】 下面通过一个实例来初步了解 Smarty 的使用过程。（**实例位置：光盘\TM\sl\21\1**）

（1）新建一个程序目录，存放位置为“服务器地址/tm/sl/21/”，命名为 1，表示为第一个实例。

（2）复制 Smarty 到目录 1 下，在 Smarty 目录下新建 4 个目录，分别是 templates、templates_c、configs 和 cache。这时，例 21.1 的目录结构如图 21.1 所示。

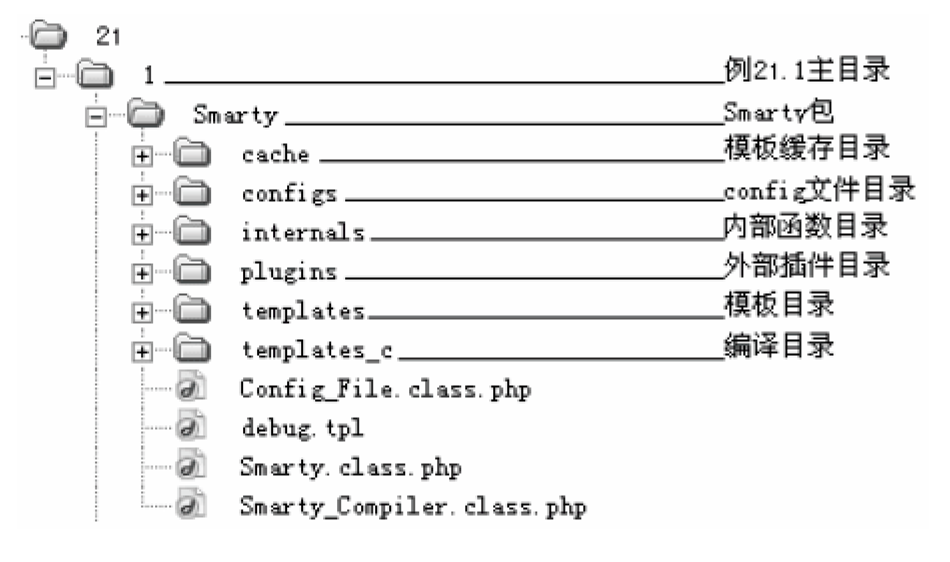


图 21.1　Smarty 包的目录结构

（3）新建一个.html 静态页，输入数据。输入完毕后将文件保存到刚新建的 templates 目录下，并命名为 index.html，实例代码如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>{ $title }</title>
</head>
<body>
{$content}
</body>
</html>
```


说明

代码中加粗的部分就是 Smarty 标签，大括号“{}”为标签的定界符，$title 和$content 为变量。21.3 节中将会详细介绍。


技巧

这里使用.html 作为模板文件的后缀，因为 HTML 网页在互联网中更容易被搜索引擎搜索到。

（4）回到上级目录，在“目录/1/”下新建一个.php 文件，使用 Smarty 变量和方法对文件进行操作，输入完毕后保存为 index.php，实例代码如下：

```
<?php
/*   定义服务器的绝对路径   */
define('BASE_PATH',$_SERVER['DOCUMENT_ROOT']);
/*   定义 Smarty 目录的绝对路径   */
define('SMARTY_PATH','\tm\21\1\Smarty\\');
/*   加载 Smarty 类库文件      */
require BASE_PATH.SMARTY_PATH.'Smarty.class.php';
/*   实例化一个 Smarty 对象   */
$smarty = new Smarty;
/*   定义各个目录的路径      */
$smarty->template_dir = BASE_PATH.SMARTY_PATH.'templates/';
$smarty->compile_dir = BASE_PATH.SMARTY_PATH.'templates_c/';
$smarty->config_dir = BASE_PATH.SMARTY_PATH.'configs/';
$smarty->cache_dir = BASE_PATH.SMARTY_PATH.'cache/';

/*   使用 Smarty 赋值方法将一对名称/方法发送到模板中   */
$smarty->assign('title','第一个 Smarty 程序');
$smarty->assign('content','Hello,Welcome to study \'Smarty\'!');
/*   显示模板   */
$smarty->display('index.html');
?>
```

这一步是 Smarty 运行最关键的核心步骤，主要进行了两项设置和两步操作。



- ☑ 加载 Smarty 类库，也就是加载 Smarty.class.php 文件，这里使用的是绝对地址。为了稍后在配置其他路径时不用输入那么长的地址字串，之前还声明了两个常量：服务器地址常量和 Smarty 路径常量，两个常量连接起来就是 Smarty 类库所在的目录。
- ☑ 保存新建的 4 个目录的绝对路径到各自的变量：在第（2）步时，曾创建了 4 个目录，这 4 个目录各有各的用途，如果没有配置目录的地址，那么服务器默认的路径就是当前执行文件所在的路径。除了上面两项必须设置的变量外，还可以改变很多 Smarty 参数值，如开启/关闭缓存、改变 Smarty 的默认定界符等，这些变量将在 21.4.2 节中介绍。
- ☑ 给模板赋值：设置成功后，需要给指定的模板赋值。assign()就是赋值方法。
- ☑ 显示模板：一切操作结束后，调用 display()方法来显示页面。实际上，用户真正看到的页面是 templates 模板目录下的 index.html 模板文件，而作为首页的 index.php，只是用来传递结果和显示模板。

打开 IE 浏览器，运行 index.php 文件。运行结果如图 21.2 所示。

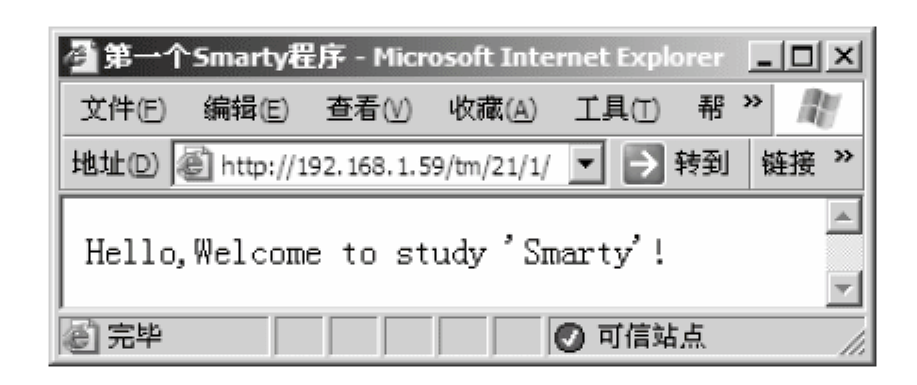


图 21.2　第一个 Smarty 程序

21.2.3　Smarty 配置

下面详细讲解 Smarty 模板的配置步骤：

（1）确定 Smarty 目录的位置。因为 Smarty 类库是通用的，每一个项目都可能会使用到它，所以将 Smarty 存储在根目录下。

（2）新建 4 个目录 templates、templates_c、configs 和 cache。其中目录 templates 存储项目的模板文件，该目录具体放置在什么位置没有严格的规定，只要设置的路径正确即可；目录 templates_c 存储项目的编译文件；目录 configs 存储项目的配置文件；目录 cache 存储项目的缓存文件。

（3）创建配置文件。如果要应用 Smarty 模板，就一定要包含 Smarty 类库和相关信息。将配置信息写到一个文件中，使用时只要 include 配置文件即可。配置文件 config.php 的代码如下：

```
<?php
/*  定义服务器的绝对路径  */
define('BASE_PATH',$_SERVER['DOCUMENT_ROOT']);
/*  定义 Smarty 目录的绝对路径  */
define('SMARTY_PATH','\mr\09\093\sl\Smarty\\');
/*  加载 Smarty 类库文件  */
require BASE_PATH.SMARTY_PATH.'Smarty.class.php';
/*  实例化一个 Smarty 对象  */
$smarty = new Smarty;
/*  定义各个目录的路径  */
$smarty->template_dir = BASE_PATH.SMARTY_PATH.'templates/';
```

```
$smarty->compile_dir = BASE_PATH.SMARTY_PATH.'templates_c/';
$smarty->config_dir = BASE_PATH.SMARTY_PATH.'configs/';
$smarty->cache_dir = BASE_PATH.SMARTY_PATH.'cache/';
?>
```

上述配置文件的参数说明如下：

☑ BASE_PATH：指定服务器的绝对路径。

技巧

指定服务器绝对路径的目的是找到 Smarty 文件夹在服务器中的存储位置。这里有两种方法可以使用：第一种，直接指定绝对路径，例如：F:\AppServ\www\\；使用这种方法来指定服务器的绝对路径，一旦服务器的绝对路径发生更改，就必须要修改配置文件，否则程序就会运行出错。第二种，通过全局变量$_SERVER['DOCUMENT_ROOT']来获取服务器的绝对路径，使用该方法不会因为服务器路径的更改而影响到程序的执行。推荐使用第二种方法定义服务器的绝对路径。

☑ SMARTY_PATH：指定 Smarty 目录的绝对路径。
☑ require：加载 Smarty 类库文件 Smarty.class.php。
☑ $smarty：实例化 Smarty 对象。
☑ $smarty->template_dir：定义模板目录存储位置。
☑ $smarty-> compile_dir：定义编译目录存储位置。
☑ $smarty-> config_dir：定义配置文件存储位置。
☑ $smarty-> cache_dir：定义模板缓存目录。

有关定界符的使用，开发者可以指定任意的格式，也可以不指定定界符，使用 Smarty 默认的定界符“{”和“}”。

至此，Smarty 的配置讲解完毕。

（4）assign 方法。

assign 用于在模板被执行时为模板变量赋值。语法格式如下：

```
{assign var=" " value=" "}
```

参数 var 是被赋值的变量名，参数 value 是赋给变量的值。

（5）display 方法。

display 方法用于显示模板，需要指定一个合法的模板资源的类型和路径。还可以通过第二个可选参数指定一个缓存号，相关的信息可以查看缓存。

```
void display (string template [, string cache_id [, string compile_id]])
```

参数 template 指定一个合法的模板资源的类型和路径，参数 cache_id 为可选参数，指定一个缓存号。

参数 compile_id 为可选参数，用于指定编译号。编译号可以将一个模板编译成不同版本使用。例如，可针对不同的语言编译模板。编译号的另外一个作用是，如果存在多个$template_dir 模板目录，但只有一个$compile_dir 编译后存档目录，这时可以为每一个$template_dir 模板目录指定一个编译号，以避免相同的模板文件在编译后互相覆盖。相对于在每一次调用 display()时都指定编译号，也可以通

过设置$compile_id 编译号属性来一次性设定。

21.3　Smarty 模板设计

视频讲解：光盘\TM\lx\21\Smarty 模板设计.exe

Smarty 的特点是将用户界面和过程分离，让美工和程序员各司其职，互不干扰。这样，Smarty 类库也自然地被分成两部分来使用，即 Smarty 模板设计和 Smarty 程序设计。两部分内容既相互独立，同时也有一部分共通。本节首先来学习 Smarty 模板设计。

21.3.1　Smarty 模板文件

Smarty 模板文件是由一个页面中所有的静态元素，加上一些定界符"{…}"组成的。模板文件统一存放的位置是 templates 目录。模板中不允许出现 PHP 代码段。Smarty 模板中的所有注释、变量、函数等都要包含在定界符内。

21.3.2　注释

Smarty 中的注释和 PHP 注释类似，都不会显示在源代码当中。注释包含在两个星号"*"中间，格式如下：

```
{* 这是注释 *}
```

21.3.3　变量

Smarty 中的变量来自以下 3 个部分：

1. PHP 页面中的变量

也就是 assign()方法传过来的变量。使用方法和在 PHP 中是一样的，也需要使用"$"符号，略有不同的是对数组的读取。在 Smarty 中读取数组有两种方法：一种是通过索引获取，和 PHP 中相似，可以是一维，也可以是多维；另一种是通过键值获取数组元素，这种方法的格式和以前接触过的不太一样，其使用符号"."作为连接符。例如，有一数组$arr = array{'object' => 'book','type' => 'computer', 'unit' => '本'}，如果要想得到 type 的值，则表达式的格式应为$arr.type。这个格式同样适用于二维数组。

【例 21.2】本例将使用上述两种方法来读取数组值。实例代码如下：（**实例位置：光盘\TM\sl\21\2**）

```
templates/02/index.html 文件
<html>
<head>
```

```
{*  页面的标题变量$title  *}
<title>{ $title }</title>
</head>
<body>
购书信息：<p>
{*  使用索引取得数组的第一个元素值  *}
图书类别：{ $arr[0] }<br />
{*  使用键值取得第二个数组元素值  *}
图书名称：{ $arr.name }<br />
{*  使用键值取得二维数组的元素值  *}
图书单价：{ $arr.unit_price.price }/{ $arr.unit_price.unit }
</body>
</html>
index.php 文件
<?php
/*  载入配置文件  */
    include '../config.php';
/*  声明数组  */
    $arr = array('computerbook','name' => 'PHP 从入门到精通','unit_price' => array('price' => '￥65.00','unit' =>
'本'));
/*  将标题和数组传递给模板  */
    $smarty->assign('title','使用 Smarty 读取数组');
    $smarty->assign('arr',$arr);
/*  要显示的模板页面  */
    $smarty->display('02/index.html');
?>
```

运行结果如图 21.3 所示。



图 21.3　使用 Smarty 读取数组

2．保留变量

相当于 PHP 中的预定义变量。在 Smarty 模板中使用保留变量时无须使用 assign()方法传值，而只需直接调用变量名即可。Smarty 中常用的保留变量如表 21.1 所示。

表 21.1　Smarty 中常用的保留变量

保留变量名	说　　明
get、post、server、session、cookie、request	等价于 PHP 中的$_GET、$_POST、$_SEVER、$_SESSION、$_COOKIE、$_REQUEST
now	当前的时间戳。等价于 PHP 中的 time()
const	用 const 包含修饰的为常量
config	配置文件内容变量。参见例 21.4

【例 21.3】 本例在模板文件中输出一些保留变量的值。实例代码如下：（**实例位置：光盘\TM\sl\21\3**）

```
templates/03/index.html 文件
{ *   设置标题名称   * }
<title>{ $title }</title>
<body>
{*   使用 get 变量获取 url 中的变量值(ex: http://localhost/tm/sl/21/3/index.php?type=computer)   *}
变量 type 的值是：{ $smarty.get.type }<br />
当前路径为：{ $smarty.server.PHP_SELF}<br />
当前时间为：{$smarty.now}
</body>
index.php 文件
<?php
    include '../config.php';                          //载入配置文件
    $smarty->assign('title','Smarty 保留变量');        //向模板中赋值
    $smarty->display('03/index.html');                //显示指定模板
?>
```

运行结果如图 21.4 所示。



图 21.4　Smarty 保留变量

3．从配置文件中读取数据

Smarty 模板也可以通过配置文件来赋值。对于 PHP 开发人员来说，对配置文件的使用从安装服务器就开始了，对文件的格式也有了一个初步的了解。调用配置文件中变量的格式有以下两种：

☑　使用“#”号，将变量名置于两个“#”号中间，即可像普通变量一样调用配置文件内容。

☑　使用保留变量中的$smarty_config.来调用配置文件。

【例 21.4】 本例通过上面两种方法来调用配置文件 04.conf 的内容。实例代码如下：（**实例位置：光盘\TM\sl\21\4**）

```
configs/04/04.conf 文件
title = "调用配置文件"
bgcolor = "#f0f0f0"
border = "5"
type = "计算机类"
name = "PHP 从入门到精通"
templates/04/infex.html 文件
{ config_load file="04/04.conf" }
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>{#title#}</title>
</head>
<body bgcolor="{#bgcolor#}">
<table border="{#border#}">
<tr>
	<td>{$smarty.config.type}</td>
	<td>{$smarty.config.name}</td>
</tr>
</table>
</body>
</html>
```

index.php 文件

```
<?php
	include_once '../config.php';
	$smarty->display('04/index.html');
?>
```

运行结果如图 21.5 所示。



图 21.5　调用配置文件

21.3.4　修饰变量

在 21.3.3 节中学习了如何在 Smarty 模板中调用变量，但有时不仅要取得变量值，还要对变量进行处理。变量修饰的一般格式如下：

```
{variable_name|modifer_name: parameter1:…}
```

☑　variable_name 为变量名称。

☑　modifer_name 为修饰变量的方法名。变量和方法之间使用符号“|”分隔。

☑　parameter1 是参数值。如果有多个参数，则使用“:”分隔开。

Smarty 提供了修饰变量的方法。常用方法和说明如表 21.2 所示。

表 21.2　修饰变量的常用方法和说明

方　法　名	说　　明
capitalize	首字母大写
count_characters:true/false	变量中的字符串个数。如果后面有参数 true，则空格也被计算；否则忽略空格
cat:“characters”	将 cat 中的字符串添加到指定字符串的后面
date_format:“%Y-%M-%D”	格式化日期和时间。等同于 PHP 中的 strftime()函数

续表

方　法　名	说　　明
default:“characters”	设置默认值。当变量为空时，将使用 default 后面的默认值
escape:“value”	用于字符串转码。value 值可以为 html、htmlall、url、quotes、hex、hexentity 和 javascript。默认为 html
lower	将变量字符串小写
nl2br	所有的换行符将被替换成 ，功能同 PHP 中的 nl2br()函数一样
regex_replace:“parameter1”:“value2”	正则替换。用 value2 替换所有符合 parameter1 标准的字串
replace:“value1”:“value2”	替换。使用 value2 替换所有 value1
string_format:“value”	使用 value 来格式化字符串。如 value 为%d，则字符串被格式化为十进制数
strip_tags	去掉所有 html 标签
upper	将变量改为大写

在对变量进行修饰时，不仅可以单独使用上面的方法，而且还可以同时使用多个。需要注意的是，在每种方法之间使用“|”分隔。

【例 21.5】 本例使用表 21.2 中的几种方法来修饰字符串。其他方法的使用读者可以自行练习。实例代码如下：**（实例位置：光盘\TM\sl\21\5）**

```
Templates/05/index.html 文件
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>{$title}</title>
<link rel="stylesheet" href="../css/style.css" />
</head>
<body>
原文：{$str}
<p>
变量中的字符数（包括空格）：{$str|count_characters:true}
<br />
使用变量修饰方法后：{$str|nl2br|upper}
</body>
</html>
index.php 文件
<?php
	include_once "../config.php";
	$str1 = '这是一个实例。';
	$str2 = "\n 图书->计算机类->php\n 书名：《PHP 从入门到精通》";
	$str3 = "\n 价格：￥59/本。";
	$smarty->assign('title','使用变量修饰方法');
	$smarty->assign('str',$str1.$str2.$str3.$str4);
	$smarty->display('05/index.html');
?>
```

运行结果如图 21.6 所示。



图 21.6　使用变量修饰方法

21.3.5　流程控制

Smarty 模板中的流程控制语句包括 if…elseif…else 条件控制语句和 foreach、section 循环控制语句。

1. If…elseif…else 语句

if 条件控制语句的使用和 PHP 中的 if 语句大同小异。需要注意的是 if 必须以/if 为结束标志。下面来看 if 语句的格式。

```
{if 条件语句 1}
      语句 1
{elseif 条件语句 2}
      语句 2
{else}
      语句 3
{/if}
```

在上述的条件语句中，除了使用 PHP 中的<、>、=、!=等常见运算符外，还可以使用 eq、ne、neq、gt、lt、lte、le、gte、ge、is even、is odd、is not even、is not odd、not、mod、div by、even by、odd by 等修饰词修饰。

【例 21.6】 在本例中，使用条件判断语句选择不同的返回信息。实例代码如下：（**实例位置：光盘\TM\sl\21\6**）

```
templates/06/index.html 文件
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>{$title}</title>
<link rel='stylesheet' href="../css/style.css" />
</head>
<body>
<p>
{if $smarty.get.type == 'tm'}
欢迎光临，{$smarty.get.type}
{else}
对不起，您不是本站 VIP，无权访问此栏目。
```



```
{/if}
</body>
</html>
index.php 文件
<?php
    include_once "../config.php";
    $smarty->assign("title","if 条件判断语句");
    $smarty->display("06/index.html");
?>
```

运行结果如图 21.7 所示。



图 21.7　if 条件判断语句

2．foreach 循环控制

Smarty 模板中的 foreach 语句可以循环输出数组。与另一个循环控制语句 section 相比，在使用格式上要简单得多，一般用于简单数组的处理。foreach 语句的使用格式如下：

```
{foreach name=foreach_name key=key item=item from=arr_name}
…
{/foreach}
```

参数含义：name 为该循环的名称；key 为当前元素的键值；item 是当前元素的变量名；from 是该循环的数组。其中，item 和 from 是必要参数，不可省略。

【例 21.7】 本例使用 foreach 语句，循环输出数组 infobook 的全部内容。实例代码如下：（**实例位置：光盘\TM\sl\21\7**）

```
templates/07/index.html 文件
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>{$title}</title>
</head>
<body>
使用 foreach 语句循环输出数组。<p>
{foreach key=key item=item from=$infobook}
{$key} => {$item}<br />
{/foreach}
</body>
</html>
index.php 文件
<?php
    include_once '../config.php';
```

```
        $infobook = array('object'=>'book','type'=>'computer','name'=>'PHP 从入门到精通','publishing'=>'清华大学出版社');
        $smarty->assign('title','使用 foreach 循环输出数组内容');
        $smarty->assign('infobook',$infobook);
        $smarty->display('07/index.html');
?>
```

运行结果如图 21.8 所示。



图 21.8　使用 foreach 循环控制输出数组内容

3．section 循环控制

Smarty 模板中的另一个循环语句是 section，该语句可用于比较复杂的数组。section 的语法结构如下：

```
{section name="sec_name"loop=$arr_name start=num step=num}
```

参数含义：name 是该循环的名称；loop 为循环的数组；start 表示循环的初始位置，例如 start=2，说明循环是从 loop 数组的第二个元素开始的；step 表示步长，例如 step=2，那么循环一次后数组的指针将向下移动两位，依此类推。

【例 21.8】 本例使用 section 语句循环输出一个二维数组。实例代码如下：（**实例位置：光盘\TM\sl\21\8**）

```
templates/08/index.html 文件
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>{$title}</title>
<link rel="stylesheet" href="../css/style.css" />
</head>
<body>
<table width="100" border="0" align="left" cellpadding="0" cellspacing="0">
{section name=sec1 loop=$obj}
    <tr>
        <td colspan="2">{$obj[sec1].bigclass}</td>
    </tr>
    {section name=sec2 loop=$obj[sec1].smallclass}
    <tr>
        <td width="25"> </td>
        <td width="75">{$obj[sec1].smallclass[sec2].s_type}</td>
    </tr>
```

```
        {/section}
{/section}
</table>
</body>
</html>
index.php 文件
<?php
        require "../config.php";
        $obj=array(array("id"=>1,"bigclass"=>"计算机图书","smallclass"=>array(array("s_id"=>1,"s_type"=>"PHP"))),
array("id"=>2,"bigclass"=>"历史传记","smallclass"=>array(array("s_id"=>2,"s_type"=>"中国历史"),array("s_id"=>3,"s_
type"=>"世界历史"))), array("id"=>3,"bigclass"=>"畅销小说","smallclass"=>array(array("s_id"=>4,"s_type"=>"网络小
说"),array("s_id" => 5, "s_type" => "科幻小说"))));
         $smarty->assign('title','section 循环控制');
        $smarty->assign("obj", $obj);
        $smarty->display("08/index.html");
?>
```

运行结果如图 21.9 所示。



图 21.9　使用 section 循环控制输出数组

21.4　Smarty 程序设计

视频讲解：光盘\TM\lx\21\Smarty 程序设计.exe

通过前面的学习已经知道，在 Smarty 模板中是不推荐使用 PHP 代码段的，所有的 PHP 程序都要另写成文件。Smarty 程序的功能主要分为两种：一种功能是和 Smarty 模板之间的交互，如方法 assign、display；另一种功能就是配置 Smarty，如变量 template_dir、$config_dir 等。本节就来学习 Smarty 程序设计的其他一些方法和配置参数。

21.4.1　Smarty 中的常用方法

Smarty 中除了使用 assign 和模板交互外，还有一些比较常用的方法。方法名称和功能说明如表 21.3 所示。

表 21.3　Smarty 程序设计常用方法和说明

方　法　名	说　　明
void append (string varname, mixed var[, boolean merge])	该方法向数组中追加元素
void clear_all_assign ()	清除所有模板中的赋值
void clear_assign (string var)	清除一个指定的赋值
void config_load (string file [, string section])	加载配置文件，如果有参数 section，说明只加载配置文件中相对应的一段数据
string fetch (string template)	返回模板的输出内容，但不直接显示出来
array get_config_vars ([string varname])	获取指定配置变量的值，如果没有参数，则返回一个所有配置变量的数组
array get_template_vars ([string varname])	获取指定模板变量的值，如果没有参数，则返回一个所有模板变量的数组
bool template_exists (string template)	检测指定的模板是否存在

这些方法在使用上和 assign、display 基本一样。这里只给出 append 方法的实例作参考，其他方法请读者自己动手实践。

【例 21.9】 本例使用 append 向数组$arr 中追加两个数组，第 3 个参数分别设为 true 和 false，查看有什么不同。实例代码如下：（**实例位置：光盘\TM\sl\21\9**）

```
templates/09/index.html 文件
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>{$title}</title>
<link rel="stylesheet" href="../css/style.css" />
</head>
<body>
{ foreach key=key item=item from=$arr }
	{$key} => {$item} <br />
{ /foreach }
</body>
</html>
index.php 文件
<?php
	include '../config.php';
	$arr = array("object"=>'book',"type"=>'computer');
	$str1 = array('name'=>'php');
	$str2 = array('publishing'=>'qinghua');
	$smarty->assign('title','使用 append');
	$smarty->assign('arr',$arr);
	$smarty->append('arr',$str1,true);
	$smarty->append('arr',$str2);
	$smarty->display('09/index.html');
?>
```

运行结果如图 21.10 所示。





图 21.10　使用 append

21.4.2　Smarty 的配置变量

Smarty 中只有一个常量 SMARTY_DIR，用来保存 Smarty 类库的完整路径，其他的所有配置信息都保存到相应的变量中。这里将介绍包括前面章节中接触过的 template_dir 等变量的作用及设置。

- ☑ $template_dir：模板目录。模板目录用来存放 Smarty 模板，在前面的实例中，所有的.html 文件都是 Smarty 模板。模板的后缀没有要求，一般为.htm、.html 等。
- ☑ $compile_dir：编译目录。顾名思义，就是编译后的模板和 PHP 程序所生成的文件默认路径为当前执行文件所在的目录下的 templates_c 目录。进入到编译目录，可以发现许多“%%…%%index.html.php”格式的文件。随便打开一个这样的文件可以发现，实际上 Smarty 将模板和 PHP 程序又重新组合成一个混编页面。
- ☑ $cache_dir：缓存目录。用来存放缓存文件。同样，在 cache 目录下可以看到生成的.html 文件。如果 caching 变量开启，那么 Smarty 将直接从这里读取文件。
- ☑ $config_dir：配置目录。该目录用来存放配置文件。例 21.4 中所用到的配置文件，就保存到这里。
- ☑ $debugging：调试变量。该变量可以打开调试控制台。只要在配置文件（config.php）中将 $smarty->debugging 设为 true 即可使用。
- ☑ $caching：缓存变量。该变量可以开启缓存。只要当前模板文件和配置文件未被改动，Smarty 就直接从缓存目录中读取缓存文件而不重新编译模板。

21.5　Smarty 模板的应用

视频讲解：光盘\TM\lx\21\Smarty 模板的应用.exe

21.5.1　将 Smarty 的配置方法封装到类中

可以将 Smarty 模板的配置方法定义到一个类中，并存储在 system.smarty.inc.php 文件中，将类的实例化操作存储到 system.inc.php 文件中，然后将这两个文件存储在 system 文件夹下。

这里将 Smarty 模板的配置存储在一个类中，通过类中的构造方法完成对 Smarty 的配置操作，这

就是 system.smarty.inc.php，其代码如下：

```
<?php
require("../Smarty/Smarty.class.php");                    //调用 Smarty 文件
class SmartyProject extends    Smarty{                    //定义类，继承 Smarty 父类
	function SmartyProject(){                             //定义方法，配置 Smarty 模板
		$this->template_dir = "./";                       //指定模板文件存储在根目录下
		$this->compile_dir = "../Smarty/templates_c/";    //指定编译文件存储在 Smarty/templates_c/文件夹下
		$this->config_dir = "../Smarty/configs/";
		$this->cache_dir = "../Smarty/cache/";
	}
}
?>
```

在 system.inc.php 中对类进行实例化，根据返回的对象名称调用 Smarty 中的方法，返回对象名为 $smarty。其代码如下：

```
<?php
	require("system.smarty.inc.php");                     //调用类文件
	$smarty=new SmartyProject();                          //执行类的实例化操作
?>
```

将配置方法封装到类中后，无论将程序复制到哪个服务器下执行，都不需要更改服务器或 Smarty 文件的绝对路径，即可直接运行。

【例 21.10】 本例应用存储在类文件中的配置方法，使用 Smarty 中的 section 循环语句，输出数据库中的数据。**（实例位置：光盘\TM\sl\21\10）**

（1）创建 index.php 动态页文件。首先，连接数据库，调用 Smarty 配置文件，通过 MySQL 数据库函数读取数据库中的数据，并把读取到的数据存储到一个数组中。然后，应用 Smarty 中的 assign 方法将数组赋给指定的模板变量。最后，使用 Smarty 中的 display 方法指定模板页。

```
<?php
	include_once "conn/conn.php";                         //连接数据库
	require_once("system/system.inc.php");                //调用指定的文件
	$result=mysql_query("select * from tb_book where id order by id limit 3",$conn);      //执行 select 查询语句
	$array=array();                                       //定义空数组
	while($myrow=mysql_fetch_array($result)){
		array_push($array,$myrow);                        //将读取到的数据写入到数组中
	}
	if(!$array){
		$smarty->assign("iscommo","F");                   //判断如果执行失败，则输出模板变量 iscommo 的值为 F
	}else{
		$smarty->assign("iscommo","T");                   //判断如果执行成功，则输出模板变量 iscommo 的值为 T
	$smarty->assign("arraybook",$array);                  //定义模板变量 arraybook，输出数据库中的数据
	}
	$smarty->display('index.html');                       //执行模板文件
?>
```

（2）创建 index.html 模板页。应用 Smarty 中的 section 循环语句，读取模板变量中的数据，在模



板页中输出从数据库中获取的数据。其关键代码如下：

```
{section name=bookid loop=$arraybook}
<tr>
    <td width="135" rowspan="5" align="center" valign="middle">
    <img src="{$arraybook[bookid].pics}" width="95" height="100" alt="{$arraybook[bookid].name}" style="border:
1px solid #f0f0f0;" /></td>
     <td height="35">图书名称：{$arraybook[bookid].name}</td>
</tr>
<tr>
    <td height="23">图书品牌：{$arraybbstell[bookid].brand}</td>
</tr>
<tr>
    <td width="160" height="23">剩余数量：{$arraybbstell[bookid].stocks}</td>
</tr>
<tr>
    <td height="23">市场价：<font color="red">{$arraybbstell[bookid].m_price} 元</font></td>
</tr>
<tr>
    <td height="30">会员价格：<font color="#FF0000">{$arraybbstell[bookid].v_price} 元</font></td>
</tr>
{/section}
```

运行结果如图 21.11 所示。



图 21.11　将 Smarty 的配置方法封装到类中

21.5.2　Smarty+ADODB 整合应用

下面介绍综合运用 Smarty 和 ADODB 技术，通过面向对象的方法完成 Smarty 模板的配置、ADODB

连接、操作 MySQL 数据库和分页的功能。

有关 ADODB 连接、操作 MySQL 和分页功能已经在第 19 章中进行了详细的讲解，这里不再赘述。

【例 21.11】 本例应用 ADODB 连接操作 MySQL 数据库，应用分页类完成数据的分页输出，应用 Smarty 模板实现网页的动静分离。**（实例位置：光盘\TM\sl\21\11）**

（1）在 system 文件夹下，创建 system.class.inc.php 文件，定义数据库的连接、操作和分页类；创建 system.smarty.inc.php 文件，定义 Smarty 的配置类；创建 system.inc.php 文件，完成类的实例化操作，并返回实例化对象和数据库的连接标识。代码可参考本书光盘中的内容。

（2）创建 index.php 动态页。调用数据库连接类中的方法完成与数据库的连接，应用分页类中的方法，实现分页读取数据库中的数据，应用 Smarty 中的 assign 方法将从数据库中读取的数据赋给模板变量，最后应用 display 方法指定模板页。其代码如下：

```
<?php
require_once("system/system.inc.php");    //调用指定的文件
$shopping=$seppage->ShowDate("select * from tb_book where id order by id ",$conn,3,$_GET["page"]);
        //调用分页类，实现分页功能
if(!$shopping){
    $smarty->assign("istr","F");
}else{
    $smarty->assign("istr","T");
     $smarty->assign("showpage",$seppage->ShowPage("图书","本","","a1"));  //定义输出分页数据的模板变量
showpage
    $smarty->assign("shopping",$shopping);          //将返回的数组赋给模板变量
}
     $smarty->assign('title','Smarty+Adodb 完成数据分页显示');
     $smarty->display('index.html');                    //指定模板页

?>
```

（3）创建 index.html 静态页，应用 section 循环语句，循环输出模板变量中传递的数据，并输出分页超链接。其关键代码如下：

```
{if $istr=="T"}
<table width="380" height="134" border="0" cellspacing="0" cellpadding="0">
{php}
        $i=1;
{/php}
{section name=shopping_id loop=$shopping}
        <tr>
                <td width="135" rowspan="5" align="center" valign="middle"><img src="{$shopping[shopping_id].pics}"
width="95" height="100" alt="{$shopping[shopping_id].name}" style="border: 1px solid #f0f0f0;" /></td>
                <td height="35">图书名称：{$shopping[shopping_id].name}</td>
        </tr>
        <tr>
                <td height="23">图书品牌：{$shopping[shopping_id].brand}</td>
        </tr>
```

```
        <tr>
            <td width="160" height="23">剩余数量：{$shopping[shopping_id].stocks}</td>
        </tr>
        <tr>
            <td height="23">市场价：<font color="red">{$shopping[shopping_id].m_price} 元</font></td>
        </tr>
        <tr>
            <td height="30">会员价格：　{$shopping[shopping_id].v_price} 元</td>
        </tr>
{php}
        $i++;
{/php}
{/section}
  </table>
<table width="100%" height="22" border="0" align="center" cellpadding="0" cellspacing="0">
        <tr>
            <td align="center" class="STYLE4"> {$showpage}</td>
        </tr>
</table>
     <hr style="border: 1px solid #f0f0f0;" />
{/if}
```

运行结果如图 21.12 所示。



图 21.12　Smarty+ADODB 整合应用

21.6　小　　结

本章主要介绍了 Smarty 模板安装、配置及使用。在使用上，Smarty 分为模板设计和程序设计。作

为一个开发者，两方面都要牢牢掌握。作为当今流行的模板，Smarty 和其他一些主流技术，如 ADODB、Ajax 等都能够很好地合作。希望读者通过本章的学习，能基本掌握 Smarty 的使用，为后面的学习做好铺垫。

21.7 练习与实践

1．truncate 方法截取字符串。**（答案位置：光盘\TM\sl\21\12）**
2．register_function 方法注册模板函数。**（答案位置：光盘\TM\sl\21\13）**

第22章

PHP与XML技术

（视频讲解：29分钟）

XML语言是目前日趋流行的语言，被称为“第二代Web语言”，是Web 2.0中的一项重要技术。无论是RSS订阅、Web Service，还是Ajax无刷新技术，都和XML有着直接的联系。通过PHP可以对XML进行全面的操作。

通过阅读本章，您可以：

- 了解XML基础知识
- 掌握使用SimpleXML解析XML文档的方法
- 掌握遍历XML文档的方法
- 掌握修改、保存XML文档的方法
- 掌握动态创建XML文档的方法

22.1　XML 的概述

XML（eXtensible Markup Language），扩展性标记语言，它是用来描述其他语言的语言，它允许用户设计自己的标记。XML 是由 W3C（World Wide Web Consortium，互联网联合组织）于 1998 年 2 月发布的一种标准，它的前身是 SGML（Standard Generalized Markup Language，标准通用标记语言）。XML 产生的原因是为了补充 HTML 语言的不足，使网络语言更加规范化、多样化。

HTML 语言被称为第一代 Web 语言，现在的版本为 4.0，以后将不再更新，取而代之的是 XHTML。而 XHTML 正是根据 XML 来制定的。XML 特点有以下几个方面。

☑ 易用性：XML 可以使用多种编辑器来进行编写，包括记事本等所有的纯文本编辑器。

☑ 结构性：XML 是具有层次结构的标记语言，包括多层的嵌套。

☑ 开放性：XML 语言允许开发人员自定义标记，这使得不同的领域都可以有自己的特色方案。

☑ 分离性：XML 语言将数据样式和数据内容分开保存、各自处理，使得基于 XML 的应用程序可以在 XML 文件中准确高效地搜索相关的数据内容，忽略其他不相关部分。

22.2　XML 语法

视频讲解：光盘\TM\lx\22\XML 的语法.exe

XML 语法是 XML 语言的基础，是学好 XML 的前提条件。任何一门语言都有一些共同的特性，同样也有各自的语法特点。下面就来学习 XML 语法特点。

22.2.1　XML 文档结构

【例 22.1】 在开始讲解 XML 语法之前，先来熟悉一下 XML 的文档结构。实例代码如下：**（实例位置：光盘\TM\sl\22\1）**

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<?xml-stylesheet type="text/css" href="Book.css"?>
<!--  下面的标签<计算机图书>就是这个 XML 文档的根目录  -->
<计算机图书>
	<PHP>
		<书名>PHP 程序开发范例宝典</书名>
		<价格  单位="元/本">89.00</价格>
		<出版时间>2007-09-01</出版时间>
	</PHP>
</计算机图书>
```

例 22.1 包含了一个 XML 文档最基本的要素，包括 XML 声明、处理指令（PI）、注释和元素等，



下面就来一一说明。

22.2.2　XML 声明

XML 声明在文档中只能出现一次，而且必须是在第一行。XML 声明包括 XML 版本、编码等信息。例 22.1 中的第一行就是该文档的声明。

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
```

XML 声明的各部分含义如表 22.1 所示。

表 22.1　XML 声明的各部分含义

XML 声明部分	含　义
<?xml	表示 XML 声明的开始。xml 表示该文件是 XML 文件
version="1.0"	XML 的版本说明，是声明中必不可少的属性，而且必须放到第一位
encoding="gb2312"	编码声明。如果不声明该属性，那么 XML 默认使用 UTF-8 来解析文档
standalone="yes"	独立声明。如果该属性赋值 yes，那么说明该 XML 文档不依赖于外部文档；如果该属性赋值为 no，则说明该文档有可能依赖于某个外部文档
?>	XML 声明的结束标记

22.2.3　处理指令

顾名思义，就是如何处理 XML 文档的指令。有一些 XML 分析器可能对 XML 文档的应用程序不做处理，这时可以指定应用程序按照这个指令信息来处理，然后再传给下一个应用程序。XML 声明其实就是一个特殊的处理指令。处理指令的格式为：

```
<?处理指令名  处理执行信息?>
```

例 22.1 中的处理指令是：

```
<?xml-stylesheet type = "text/css" href="Book.css"?>
```

具体含义如下。

- ☑　xml-stylesheet：样式表单处理指令，指明了该 XML 文档所使用的样式表。
- ☑　type="text/css"：设定了文档所使用的样式是 css。
- ☑　href="Book.css"：设定了样式文件的地址。

22.2.4　注释

XML 中的注释和 HTML 是一样的，使用“<!--”和“-->”作为开始和结束界定符。注释的用法十分简单，这里只介绍在使用注释时要注意的几个问题。

- ☑ 不能出现在 XML 声明之前。
- ☑ 不能出现在 XML 元素中间。如<computer_book <!-- 这是错误的 -->>。
- ☑ 不能出现在属性列表中。
- ☑ 不能嵌套注释。
- ☑ 注释内容可以包含“<”、“>”和“&”等特殊字符，但不允许有“--”。

22.2.5　XML 元素

元素是每个 XML 文档不可或缺的部分，也是文档内容的基本单元。每个 XML 文档至少要包含一个元素。一般元素由 3 部分组成，格式如下：

```
<标签>数据内容</标签>
```

其中，<标签>为元素的开始标签，</标签>是元素的结束标签，中间的数据内容是元素的值。这里要注意的是标签的写法。

- ☑ <标签>和</标签>都是成对出现的，这是 XML 严格定义的，不允许只有开始标签而没有结束标签，对于空元素，即两个标签之间没有数据，这时可以使用简短形式<标签/>。
- ☑ 英文标签名称只能由下划线“_”或英文字母开头，中文标签名称只能使用下划线“_”或汉字开头。名称中只能有下划线“_”、连接符“-”、点“.”和冒号“:”等特殊字符，也可以使用指定字符集下的合法字符。
- ☑ <标签>中不能有空格，< 标签>或</ 标签>都是错误的。
- ☑ <标签>对英文大小写敏感，如<name>和<Name>是两个不同的标签。

22.2.6　XML 属性

XML 属性是 XML 元素中的内容，是可选的。XML 属性和 HTML 中的属性在功能上十分相似，但 XML 属性在格式上更加严格，使用上更加灵活。XML 属性的格式为：

```
<标签 属性名="属性值" 属性名=""…>内容</标签>
```

这里要注意：

- ☑ 属性名和属性值必须是成对出现的，不像 HTML 中有些属性，可以不需要值而单独存在。对于 XML 来说这是不允许的。如果没有值，写成“属性名=""”也可以。
- ☑ 属性值必须用引号括起来，通常使用双引号，除非属性值本身包含了双引号，这时可以用单引号来代替。

22.2.7　使用 CDATA 标记

在 XML 中，特殊字符“>”、“<”和“&”的输入需要使用实体引用来处理，实体引用就是使用



"&…;"的形式来代替那些特殊字符。表 22.2 是 XML 中所用到的实体引用。

表 22.2　XML 中的实体引用

实 体 参 考	字　符
<	<
>	>
'	'
"	"
&	&

但如果遇到大量的特殊符号需要输入，使用这种方法就不太实际了。XML 中提供了 CDATA（Character data，字符数据）标记，在 CDATA 标记段的内容都会被当作纯文本数据处理。CDATA 标记的格式如下：

```
<![CDATA[
…
]]>
```

【例 22.2】 在本例中，分别使用实体引用和 CDATA 标记来显示特殊符号。实例代码如下：（**实例位置：光盘\TM\sl\22\2**）

```
<?xml version="1.0" encoding="GB2312"?>
<exam>
      <实体引用>这里必须使用引用"&lt;"、"&gt;"、"&amp;"</实体引用>
      <CDATA 标记>
      <![CDATA[
            这里可以正常输出"<"、">"、"&"。
      ]]>
      </CDATA 标记>
</exam>
```

注意

在 CDATA 标记段内不允许出现"]]>"，否则，XML 会认为 CDATA 标记段结束。

22.2.8　XML 命名空间

命名空间通过在元素前面增加一个前缀来保证元素和属性的唯一性，它的最重要用途是用于融会不同的 XML 文档。命名空间的格式为：

```
<标签名称  xmlns:前缀名称="URL">
```

【例 22.3】 本例对元素<外语图书>使用命名空间。实例代码如下：（**实例位置：光盘\TM\sl\22\3**）

```
<?xml version="1.0" encoding="gb2312" standalone="yes"?>
<外语图书  xmlns:frn="http://www.bccd.com/foreign">
```

```
        <frn:English>
            <frn:书名>许国璋英语</frn:书名>
            <frn:价格  货币种类="RMB"  单位="4 本">80.00</frn:价格>
            <frn:出版时间>1996-05-10</frn:出版时间>
        </frn:English>
</外语图书>
```

22.3　在 PHP 中创建 XML 文档

视频讲解：光盘\TM\lx\22\在 PHP 中创建 XML 文档.exe

PHP 不仅可以生成动态网页，同样也可以生成 XML 文件。下面就来介绍 PHP 是如何生成 XML 的。

【例 22.4】 本例中，输出一个简单的 XML 文档。可以看到，在 PHP 中生成 XML 非常简单。实例代码如下：**（实例位置：光盘\TM\sl\22\4）**

```
<?php
        header('Content-type:text/xml');
        echo '<?xml version="1.0"    encoding="gb2312" ?>';
        echo '<计算机图书>';
        echo '<PHP>';
        echo '<书名>PHP 项目开发全程实录</书名>';
        echo '<价格>85.00RMB</价格>';
        echo '<出版日期>2008-5-5</出版日期>';
        echo '</PHP>';
        echo '</计算机图书>';
?>
```

运行结果如图 22.1 所示。

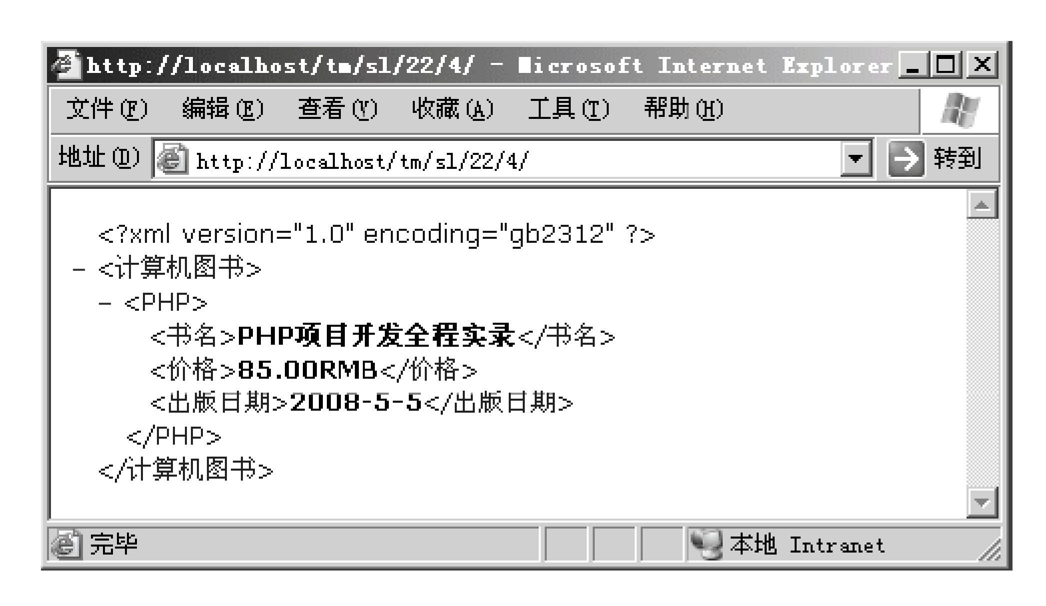


图 22.1　在 PHP 中创建 XML 文件

22.4　SimpleXML

视频讲解：光盘\TM\lx\22\SimpleXML 类库.exe

PHP 对 XML 格式的文档进行操作有很多方法。如 XML 语法解析函数、DOMXML 函数和 SimpleXML

函数等。其中，PHP 5 新加入的 SimpleXML 函数操作更简单。本节就使用 SimpleXML 系列函数来实现对 XML 文档的读写和浏览。

22.4.1　创建 SimpleXML 对象

使用 SimpleXML 首先要创建对象。共有 3 种方法来创建对象，分别是：

☑ Simplexml_load_file()函数，将指定的文件解析到内存中。

☑ Simplexml_load_string()函数，将创建的字符串解析到内存中。

☑ Simplexml_load_date()函数，将一个使用 DOM 函数创建的 DomDocument 对象导入到内存中。

【例 22.5】 本例使用 3 个函数分别创建 3 个对象，并使用 print_r 来输出 3 个对象。实例代码如下：**（实例位置：光盘\TM\sl\22\5）**

```
<?php
header("Content-Type:text/html;charset=utf-8");                    //设置编码
/*  第一种方法  */
$xml_1 = simplexml_load_file("5.xml");
print_r($xml_1);
/*  第二种方法  */
$str = <<<XML
<?xml version='1.0' encoding='gb2312'?>
<Object>
	<ComputerBook>
		<title>PHP 从入门到精通</title>
	</ComputerBook>
</Object>
XML;
$xml_2 = simplexml_load_string($str);
echo '<p>';
print_r($xml_2);
/*  第三种方法  */
$dom = new domDocument();
$dom -> loadXML($str);
$xml_3 = simplexml_import_dom($dom);
echo '<p>';
print_r($xml_3);
?>
```

结果为：

```
SimpleXMLElement Object ( [ComputerBook] => SimpleXMLElement Object ( [title] => PHP 从入门到精通 ) )
SimpleXMLElement Object ( [ComputerBook] => SimpleXMLElement Object ( [title] => PHP 从入门到精通 ) )
SimpleXMLElement Object ( [ComputerBook] => SimpleXMLElement Object ( [title] => PHP 从入门到精通 ) )
```

可以看到，不同数据源的 XML 只要结构相同，那么输出的结果也是相同的。


注意

第一行中的 header()函数设置了 html 编码。虽然在 XML 文档中设置了编码格式，但只是针对 XML 文档的，在 html 输出时也要设置编码格式。

22.4.2 遍历所有子元素

创建对象后，就可以使用 SimpleXML 的其他函数来读取数据了。使用 SimpleXML 对象中的 children()函数和 foreach 循环语句可以遍历所有子节点元素。

【例 22.6】本例使用 children()函数遍历所有子节点。实例代码如下：（**实例位置：光盘\TM\sl\22\6**）

```
<?php
header('Content-Type:text/html;charset=utf-8');                    //设置编码
/*   创建 XML 格式的字符串   */
$str = <<<XML
<?xml version='1.0' encoding='gb2312'?>
<object>
	<book>
		<computerbook>PHP 从入门到精通</computerbook>
	</book>
	<book>
		<computerbook>PHP 项目开发全程实录</computerbook>
	</book>
</object>
XML;
/*   **************************   */
$xml = simplexml_load_string($str);                               //创建一个 SimpleXML 对象
foreach($xml->children() as $layer_one){                          //循环输出根节点
	print_r($layer_one);                                          //查看节点结构
	echo '<br>';
	foreach($layer_one->children() as $layer_two){                //循环输出第二层根节点
		print_r($layer_two);                                      //查看节点结构
		echo '<br>';
	}
}
?>
```

运行结果如图 22.2 所示。

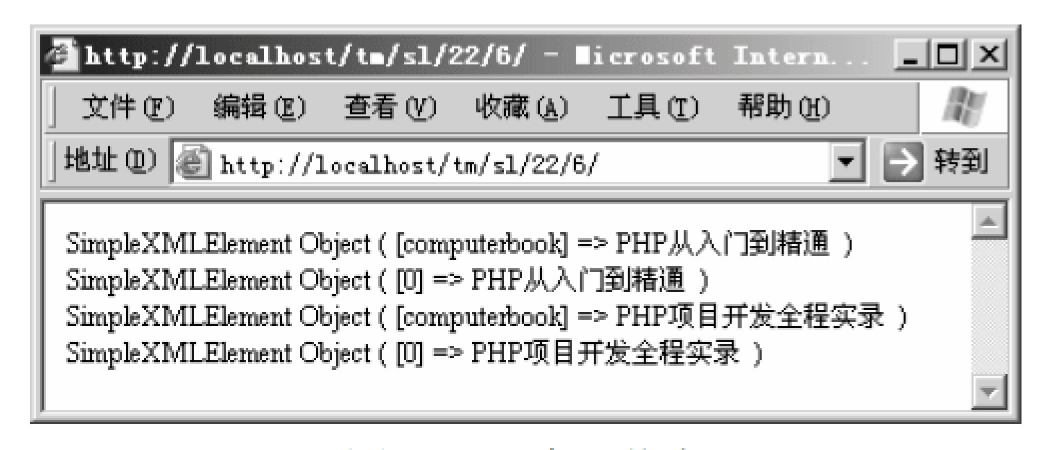


图 22.2 遍历节点



22.4.3　遍历所有属性

SimpleXML 不仅可以遍历子元素，还可以遍历元素中的属性，其使用的是 SimpleXML 对象中的 attributes()方法，在使用上和 children()相似。

【例 22.7】 本例使用 attributes()方法来遍历所有的元素属性。实例代码如下：**（实例位置：光盘\TM\sl\22\7）**

```
<?php
header("Content-Type:text/html;charset=utf-8");                    //设置编码
/*  创建 XML 格式的字符串  */
$str = <<<XML
<?xml version='1.0' encoding='gb2312'?>
<object name='commodity'>
	<book type='computerbook'>
		<bookname name='PHP 从入门到精通'/>
	</book>
	<book type='historybook'>
		<booknanme name='上下五千年'/>
	</book>
</object>
XML;
$xml = simplexml_load_string($str);                                //创建一个 SimpleXML 对象
foreach($xml->children() as $layer_one){                           //循环子节点元素
	foreach($layer_one->attributes() as $name => $vl){             //输出各个节点的属性和值
		echo $name.'::'.$vl;
	}
	echo '<br>';
	foreach($layer_one->children() as $layer_two){                 //输出第二层节点元素
		foreach($layer_two->attributes() as $nm => $vl){           //输出各个节点的属性和值
			echo $nm."::".$vl;
		}
		echo '<br>';
	}
}
?>
```

运行结果如图 22.3 所示。



图 22.3　遍历子元素属性

22.4.4　访问特定节点元素和属性

SimpleXML 对象除了可以使用上面两个方法来遍历所有的节点元素和属性，还可以访问特定的数据元素。SimpleXML 对象可以通过子元素的名称对该子元素赋值，或使用子元素的名称数组来对该子元素的属性赋值。

【例 22.8】 本例使用 SimpleXML 对象直接对 XML 元素和属性进行访问。实例代码如下：**（实例位置：光盘\TM\sl\22\8）**

```
<?php
header('Content-Type:text/html;charset=utf-8');                    //设置编码
/*   创建 XML 格式的字符串   */
$str = <<<XML
<?xml version='1.0' encoding='gb2312'?>
<object name='商品'>
        <book>
                <computerbook>PHP 从入门到精通</computerbook>
        </book>
        <book>
                <computerbook name='PHP 项目开发全程实录'/>
        </book>
</object>
XML;
/*   **************************   */
$xml = simplexml_load_string($str);                                 //创建 SimpleXML 对象
echo $xml[name].'<br>';                                             //输出根元素的属性 name
echo $xml->book[0]->computerbook.'<br>';                            //输出子元素中 computerbook 的值
echo $xml->book[1]->computerbook['name'].'<br>';                    //输出 computerbook 的属性值
?>
```

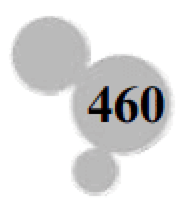
运行结果如图 22.4 所示。



图 22.4　访问特定的节点元素和属性

22.4.5　修改 XML 数据

修改 XML 数据同读取 XML 数据类似。如例 22.8，在访问特定元素或属性时，也可以对其进行修改操作。



【例 22.9】 本实例首先读取 XML 文档，然后输出根元素的属性 name，接着修改子元素 computerbook，最后输出修改后的值。实例代码如下：**（实例位置：光盘\TM\sl\22\9）**

```
<?php
/*   设置编码格式   */
header('Content-Type:text/html;charset=utf-8');
/*   创建 XML 格式的字符串   */
$str = <<<XML
<?xml version='1.0' encoding='gb2312'?>
<object name='商品'>
        <book>
                <computerbook type='PHP 入门应用'>PHP 从入门到精通</computerbook>
</book>
</object>
XML;
/*   ****************************   */
/*   创建 SimpleXML 对象   */
$xml = simplexml_load_string($str);
/*   输出根目录属性 name 的值   */
echo $xml[name].'<br />';
/*   修改子元素 computerbook 的属性值 type   */
$xml->book->computerbook['type'] = iconv('gb2312','utf-8','PHP 程序员必备工具');
/*   修改子元素 computerbook 的值   */
$xml->book->computerbook = iconv('gb2312','utf-8','PHP 函数参考大全');
/*   输出修改后的属性和元素值   */
echo $xml->book->computerbook['type'].' => ';
echo $xml->book->computerbook;
?>
```

运行结果如图 22.5 所示。

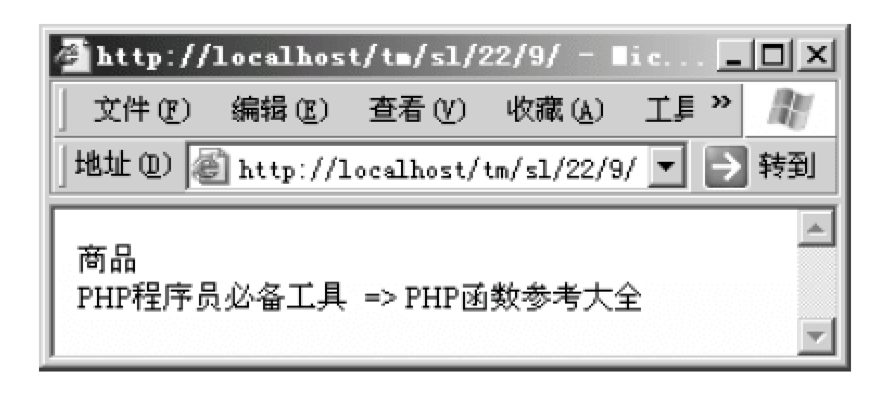


图 22.5　修改元素和属性值



说明

iconv()函数是转换编码函数。有时，希望向页面或文件写入数据，但添加的数据的编码格式和文件原有编码格式不符，导致输出时出现乱码。这时，使用 iconv()函数将数据从输入时所使用的编码转换为另一种编码格式后再输出即可解决问题。本例是将字符串“PHP 程序员必备工具”从 gb2312 的编码格式转换成 utf-8 编码格式。

22.4.6　保存 XML 文档

数据在 SimpleXML 对象中所做的修改，其实是在系统内存中做的改动，而原文档根本没有变化。当关掉网页或清空内存时，数据又会恢复。要保存一个修改过的 SimpleXML 对象，可以使用 asXML()方法来实现。该方法可以将 SimpleXML 对象中的数据格式化为 XML 格式，然后再使用 file()函数中的写入函数将数据保存到 XML 文件中。

【例 22.10】 本实例首先从 10.xml 文档中生成 SimpleXML 对象，然后对 SimpleXML 对象中的元素进行修改，最后将修改后的 SimpleXML 对象再保存到 10.xml 文档中，实例代码如下：（**实例位置：光盘\TM\sl\22\10**）

```
10.xml 文档
<?xml version="1.0" encoding="gb2312"?>
<object name="商品">
      <book>
            <computerbook type="PHP 入门应用">PHP 从入门到精通</computerbook>
      </book>
</object>
index.php 文件
<?php
/*  创建 SimpleXML 对象  */
$xml = simplexml_load_file('10.xml');
/*  修改 XML 文档内容  */
$xml->book->computerbook['type'] = iconv('gb2312','utf-8','PHP 程序员必备工具');
$xml->book->computerbook = iconv('gb2312','utf-8','PHP 函数参考大全');
/*  格式化对象$xml  */
$modi = $xml->asXML();
/*  将对象保存到 10.xml 文档中  */
file_put_contents('10.xml',$modi);
/*  重新读取 xml 文档  */
$str = file_get_contents('10.xml');
/*  输出修改后的文档内容  */
echo $str;
?>
```

运行结果如图 22.6 所示。

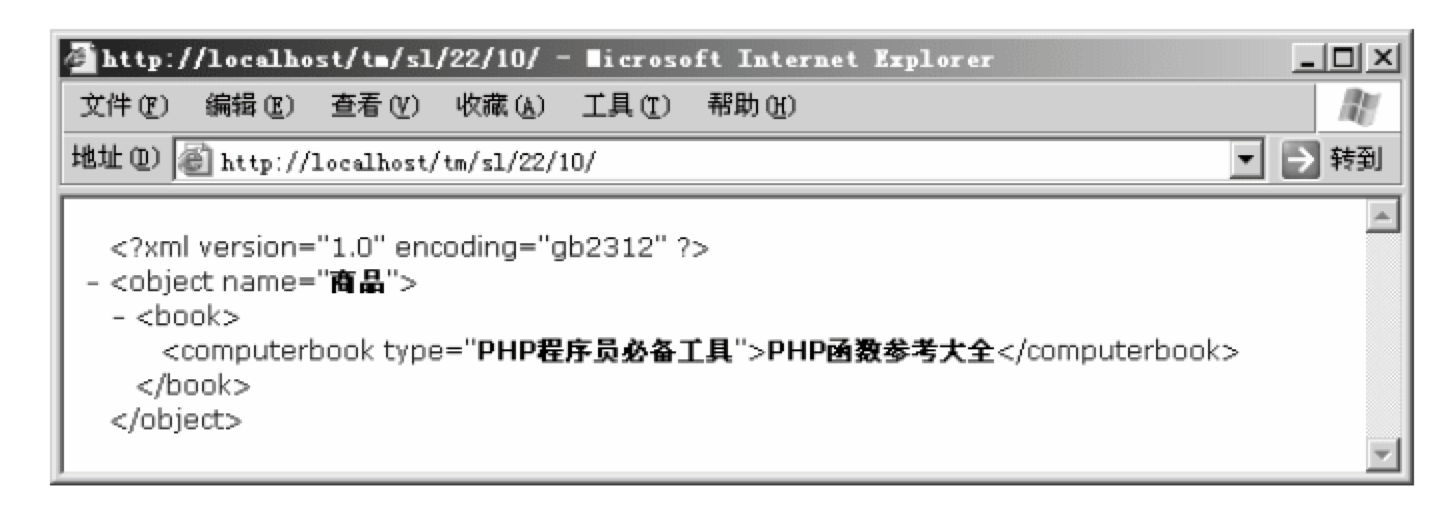


图 22.6　保存 SimpleXML 对象



22.5　动态创建 XML 文档

视频讲解：光盘\TM\lx\22\动态创建 XML 文档.exe

使用 SimpleXML 对象可以十分方便地读取和修改 XML 文档，但却无法动态建立 XML，这时就需要使用 DOM 来实现。DOM（Document Object Model），文档对象模型，通过树形结构模式来遍历 XML 文档。使用 DOM 遍历文档的好处是不需要标记即可显示全部内容，但缺点同样明显，就是十分消耗内存。

【例 22.11】 PHP 中的 DOM 函数库十分庞大，这里只给出一个常用的创建 XML 文档的实例。感兴趣的读者可以参考 XML 和 PHP 的官方手册来了解 DOM 的知识。实例代码如下：（**实例位置：光盘\TM\sl\22\11**）

```
<?php
$dom = new DomDocument('1.0','gb2312');                               //创建 DOM 对象
$object = $dom->createElement('object');                              //创建根节点 object
$dom->appendChild($object);                                           //将创建的根节点添加到 DOM 对象中
$book = $dom->createElement('book');                                  //创建节点 book
$object->appendChild($book);                                          //将节点 book 追加到 DOM 对象中
$computerbook = $dom->createElement('computerbook');                  //创建节点 computerbook
$book->appendChild($computerbook);                                    //将 computerbook 追加到 DOM 对象中
$type = $dom->createAttribute('type');                                //创建一个节点属性 type
$computerbook->appendChild($type);                                    //将属性追加到 computerbook 元素后
$type_value = $dom->createTextNode('computer');                       //创建一个属性值
$type->appendChild($type_value);                                      //将属性值赋给 type
$bookname = $dom->createElement('bookname');                          //创建节点 bookname
$computerbook->appendChild($bookname);                                //将节点追加到 DOM 对象中
$bookname_value = $dom->createTextNode(iconv('gb2312','utf-8','PHP 从入门到精通'));//创建元素值
$bookname->appendChild($bookname_value);                              //将值赋给节点 bookname
echo $dom->saveXML();                                                 //输出 XML 文件
?>
```

运行结果如图 22.7 所示。

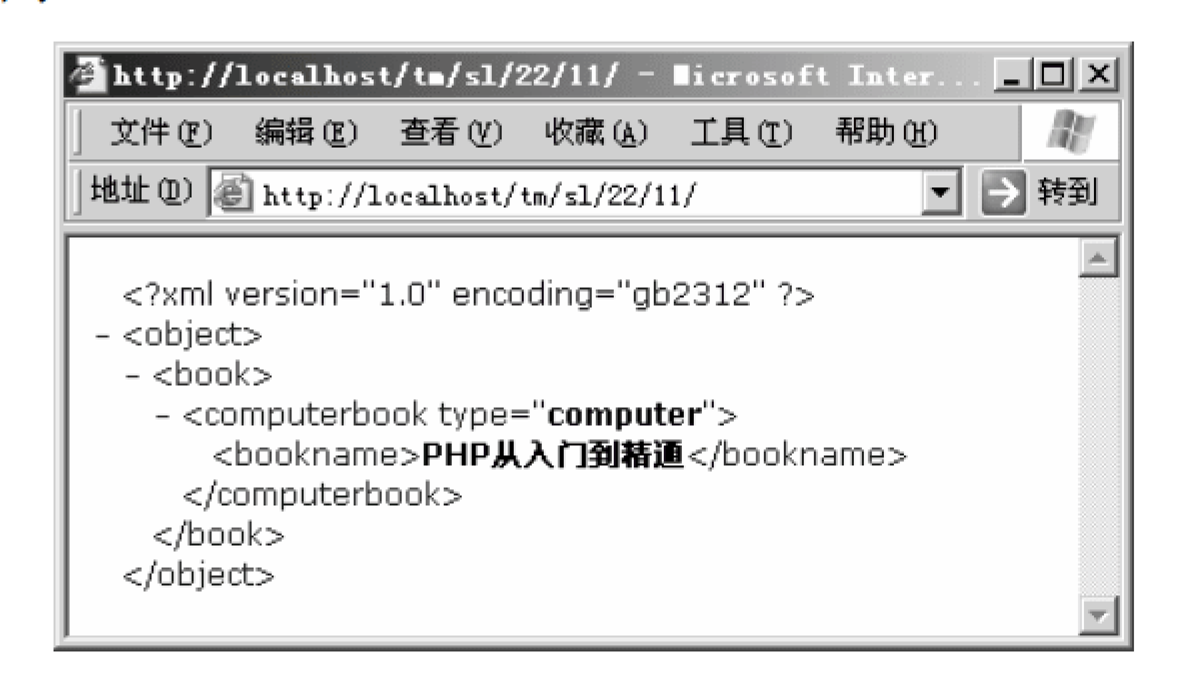


图 22.7　使用 DOM 创建 XML 文档

22.6 小 结

本章首先介绍了 XML 的基础语法，使读者对 XML 有了一个初步印象，然后学习如何在 PHP 中创建 XML 文档，接着对 PHP 5 最新的 SimpleXML 类库进行了详细的介绍，最后使用 DOM 对象模型动态创建了一个 XML 文档。

希望读者可以通过本章的概念和实例，初步掌握 PHP 对 XML 文档的操作，为学习后面的 Ajax、SOAP 等目前的流行技术做好准备。

22.7 练习与实践

1．定义一个 PHP 读取 XML 类，实现基本操作。如创建、遍历、读取、删除等。**（答案位置：光盘\TM\sl\22\12）**

2．使用 XML 来存储少量的数据。**（答案位置：光盘\TM\sl\22\13）**

第23章

PHP 与 Ajax 技术

（视频讲解：31 分钟）

随着 Web 2.0 时代的到来，Ajax 产生并逐渐成为主流。相对于传统的 Web 应用开发，Ajax 运用的是更加先进、更加标准化、更加高效的 Web 开发技术体系。需要说明的是，Ajax 是一个客户端技术，无论使用哪种服务器端技术（如 PHP、JSP、ASP 等）都可以使用 Ajax 技术。本章主要介绍 Ajax 技术及如何在 PHP 中应用 Ajax 技术。

通过阅读本章，您可以：

- 了解什么是 Ajax
- 了解 Ajax 的开发模式
- 了解 Ajax 的优点
- 掌握 Ajax 的使用技术
- 熟悉 Ajax 开发需要注意的问题
- 灵活运用 Ajax 技术在 PHP 中的应用

23.1　Ajax 概述

视频讲解：光盘\TM\lx\23\了解 Ajax.exe

Ajax 技术是目前最流行的技术，它极大地改善了传统 Web 应用的用户体验，因此也被称为传统的 Web 技术革命。Ajax 极大地发掘了 Web 浏览器的潜力，开创了大量新的可能性。下面对 Ajax 技术进行详细的介绍。

23.1.1　什么是 Ajax

Ajax 是由 Jesse James Garrett 创造的，是 Asynchronous JavaScript And XML 的缩写，即异步 JavaScript 和 XML 技术。Ajax 并不是一门新的语言或技术，它是 JavaScript、XML、CSS、DOM 等多种已有技术的组合，它可以实现客户端的异步请求操作，这样可以实现在不需要刷新页面的情况下与服务器进行通信，从而减少了用户的等待时间。

23.1.2　Ajax 的开发模式

在传统的 Web 应用模式中，页面中用户的每一次操作都将触发一次返回 Web 服务器的 HTTP 请求，服务器进行相应的处理（获得数据、运行与不同的系统会话）后，返回一个 HTML 页面给客户端。如图 23.1 所示。而在 Ajax 应用中，页面中用户的操作将通过 Ajax 引擎与服务器端进行通信，然后将返回结果提交给客户端页面的 Ajax 引擎，再由 Ajax 引擎来决定将这些数据插入到页面的指定位置。如图 23.2 所示。

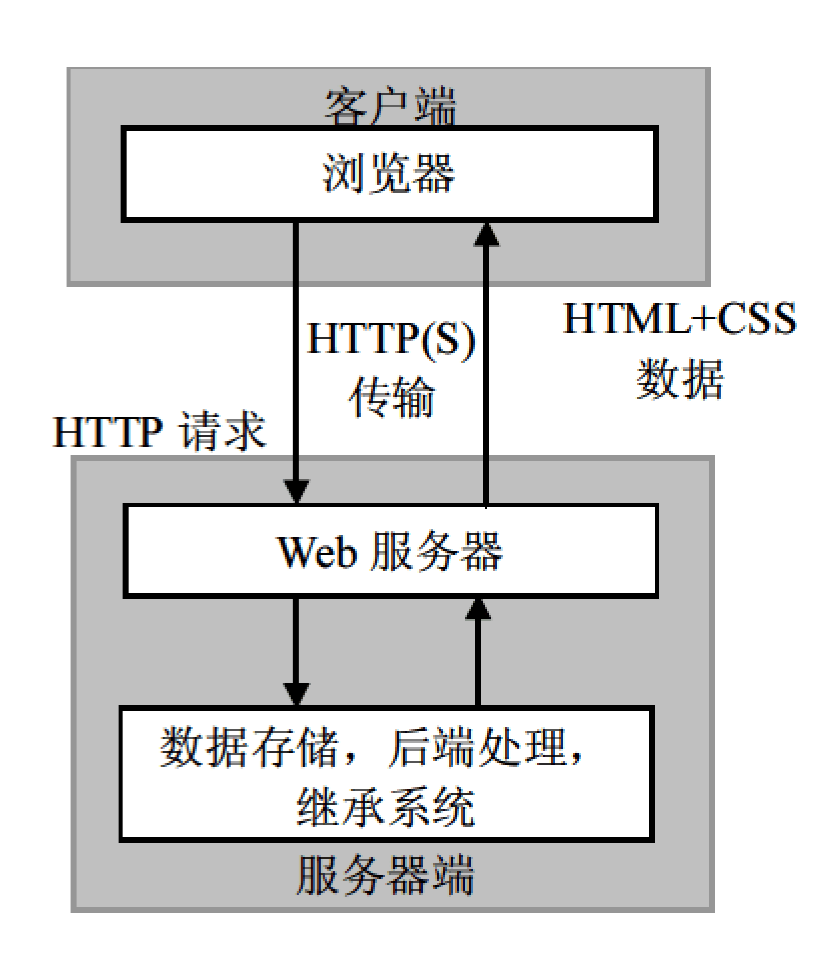


图 23.1　传统的 Web 开发模式

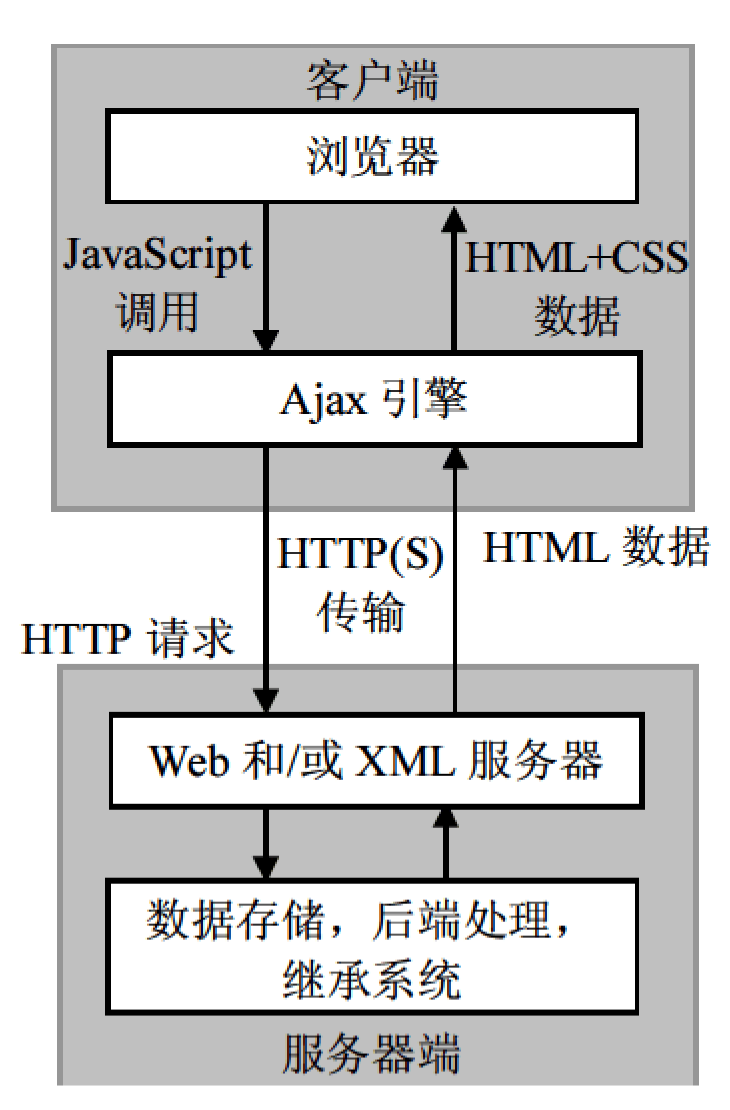
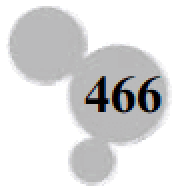


图 23.2　Ajax 的开发模式



从图 23.1 和图 23.2 中可以看出，对于每个用户的行为，在传统的 Web 应用模式中，将生成一次 HTTP 请求，而在 Ajax 应用开发模式中，将变成对 Ajax 引擎的一次 JavaScript 调用。在 Ajax 应用开发模式中通过 JavaScript 实现在不刷新整个页面的情况下，对部分数据进行更新，从而降低了网络流量，带来更好的用户体验。

23.1.3　Ajax 的优点

与传统的 Web 应用不同，Ajax 在用户与服务器之间引入一个中间媒介（Ajax 引擎），Web 页面不用打断交互流程进行重新加载即可动态地更新，从而消除了网络交互过程中的“处理—等待—处理—等待”的缺点。

使用 Ajax 的优点具体表现在以下几个方面：

☑ 减轻服务器的负担。Ajax 的原则是“按需求获取数据”，可以最大程度地减少冗余请求和响应对服务器造成的负担。

☑ 可以把一部分以前由服务器负担的工作转移到客户端，利用客户端闲置的资源进行处理，减轻服务器和带宽的负担，节约空间和宽带租用成本。

☑ 无刷新更新页面，使用户不用再像以前一样在服务器处理数据时只能在死板的白屏前焦急地等待。Ajax 使用 XMLHttpRequest 对象发送请求并得到服务器响应，在不需要重新载入整个页面的情况下，即可通过 DOM 及时将更新的内容显示在页面上。

☑ 可以调用 XML 等外部数据，进一步实现 Web 页面显示和数据的分离。

☑ 基于标准化的并被广泛支持的技术，不需要下载插件或者小程序。

23.2　Ajax 使用的技术

视频讲解：光盘\TM\lx\23\Ajax 使用的技术.exe

23.2.1　JavaScript 脚本语言

JavaScript 是一种在 Web 页面中添加动态脚本代码的解释性程序语言，其核心已经嵌入到目前主流的 Web 浏览器中。虽然平时应用最多的是通过 JavaScript 实现一些网页特效及表单数据验证等功能，但 JavaScript 可以实现的功能远不止这些。JavaScript 是一种具有丰富的面向对象特性的程序设计语言，利用它能执行许多复杂的任务，例如，Ajax 就是利用 JavaScript 将 DOM、XHTML（或 HTML）、XML 以及 CSS 等技术综合起来，并控制它们的行为。因此，要开发一个复杂高效的 Ajax 应用程序，就必须对 JavaScript 有深入的了解。关于 JavaScript 脚本语言的详细讲解可参考相关书籍。

23.2.2　XMLHttpRequest

Ajax 技术中，最核心的技术就是 XMLHttpRequest，它是一个具有应用程序接口的 JavaScript 对象，

能够使用超文本传输协议（HTTP）连接服务器，是微软公司为了满足开发者的需要，于 1999 年在 IE 5.0 浏览器中率先推出的。现在许多浏览器都对其提供了支持，但实现方式与 IE 有所不同。

通过 XMLHttpRequest 对象，Ajax 可以像桌面应用程序一样只同服务器进行数据层面的交换，而不用每次都刷新页面，也不用每次都将数据处理的工作交给服务器来做，这样既减轻了服务器负担又加快了响应速度，从而缩短了用户等待的时间。

在使用 XMLHttpRequest 对象发送请求和处理响应之前，首先需要初始化该对象，由于 XMLHttpRequest 不是一个 W3C 标准，所以对于不同的浏览器，初始化的方法也不同。

☑ IE 浏览器

IE 浏览器把 XMLHttpRequest 实例化为一个 ActiveX 对象。具体方法如下：

```
var http_request = new ActiveXObject("Msxml2.XMLHTTP");
```

或者

```
var http_request = new ActiveXObject("Microsoft.XMLHTTP");
```

在上面代码中，Msxml2.XMLHTTP 和 Microsoft.XMLHTTP 是针对 IE 浏览器的不同版本而进行设置的，目前比较常用的为这两种。

☑ Mozilla、Safari 等其他浏览器

Mozilla、Safari 等其他浏览器把它实例化为一个本地 JavaScript 对象。具体方法如下：

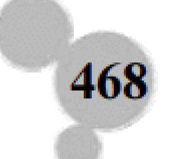
```
var http_request = new XMLHttpRequest();
```

为了提高程序的兼容性，可以创建一个跨浏览器的 XMLHttpRequest 对象。方法很简单，只需要判断一下不同浏览器的实现方式，如果浏览器提供了 XMLHttpRequest 类，则直接创建一个实例，否则使用 IE 的 ActiveX 控件。具体代码如下：

```
if (window.XMLHttpRequest) {                                    //Mozilla、Safari 等浏览器
    http_request = new XMLHttpRequest();
}
else if (window.ActiveXObject) {                                //IE 浏览器
    try {
        http_request = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (e) {
        try {
            http_request = new ActiveXObject("Microsoft.XMLHTTP");
      } catch (e) {}
    }
}
```

说明

由于 JavaScript 具有动态类型特性，而且 XMLHttpRequest 对象在不同浏览器上的实例是兼容的，所以可以用同样的方式访问 XMLHttpRequest 实例的属性或方法，不需要考虑创建该实例的方法。



下面分别介绍 XMLHttpRequest 对象的常用方法和属性。

1. XMLHttpRequest 对象的常用方法

下面对 XMLHttpRequest 对象的常用方法进行详细介绍。

（1）open()方法

open()方法用于设置进行异步请求目标的 URL、请求方法以及其他参数信息，具体语法如下：

```
open("method","URL"[,asyncFlag[,"userName"[, "password"]]])
```

在上面的语句中，method 用于指定请求的类型，一般为 get 或 post；URL 用于指定请求地址，可以使用绝对地址或者相对地址，并且可以传递查询字符串；asyncFlag 为可选参数，用于指定请求方式，同步请求为 true，异步请求为 false，默认情况下为 true；userName 为可选参数，用于指定用户名，没有时可省略；password 为可选参数，用于指定请求密码，没有时可省略。

（2）send()方法

send()方法用于向服务器发送请求。如果请求声明为异步，该方法将立即返回，否则将直到接收到响应为止。具体语法格式如下：

```
send(content)
```

在上面的语法中，content 用于指定发送的数据，可以是 DOM 对象的实例、输入流或字符串。如果没有参数，需要传递时可以设置为 null。

（3）setRequestHeader()方法

setRequestHeader()方法为请求的 HTTP 头设置值。具体语法格式如下：

```
setRequestHeader("label", "value")
```

在上面的语句中，label 用于指定 HTTP 头，value 用于为指定的 HTTP 头设置值。


注意

setRequestHeader()方法必须在调用 open()方法之后才能调用。

（4）abort()方法

abort()方法用于停止当前异步请求。

（5）getAllResponseHeaders()方法

getAllResponseHeaders()方法用于以字符串形式返回完整的 HTTP 头信息，当存在参数时，表示以字符串形式返回由该参数指定的 HTTP 头信息。

2. XMLHttpRequest 对象的常用属性

XMLHttpRequest 对象的常用属性如表 23.1 所示。

表 23.1　XMLHttpRequest 对象的常用属性

属　　性	说　　明
onreadystatechange	每个状态改变时都会触发这个事件处理器，通常会调用一个 JavaScript 函数
readyState	请求的状态。有以下 5 个取值： 0=未初始化 1=正在加载 2=已加载 3=交互中 4=完成
responseText	服务器的响应，表示为字符串
responseXML	服务器的响应，表示为 XML。这个对象可以解析为一个 DOM 对象
status	返回服务器的 HTTP 状态码，如： 200="成功" 202="请求被接收，但尚未成功" 400="错误的请求" 404="文件未找到" 500="内部服务器错误"
statusText	返回 HTTP 状态码对应的文本

23.2.3　XML 语言

XML 是 eXtensible Markup Language（可扩展的标记语言）的缩写，它提供了用于描述结构化数据的格式。XMLHttpRequest 对象与服务器交换的数据，通常采用 XML 格式，但也可以是基于文本的其他格式。

23.2.4　DOM

DOM 是 Document Object Model（文档对象模型）的简称，它为 XML 文档的解析定义了一组接口。解析器读入整个文档，然后构建一个驻留内存的树结构，最后通过 DOM 可以遍历树以获取来自不同位置的数据，可以添加、修改、删除、查询和重新排列树及其分支。另外，还可以根据不同类型的数据源来创建 XML 文档。在 Ajax 应用中，通过 JavaScript 操作 DOM，可以达到在不刷新页面的情况下实时修改用户界面的目的。

23.2.5　CSS

CSS 是 Cascading Style Sheet（层叠样式表）的缩写，用于控制网页样式并允许将样式信息与网页内容分离的一种标记性语言。在 Ajax 中，通常使用 CSS 进行页面布局，并通过改变文档对象的 CSS 属性控制页面的外观和行为。CSS 是 Ajax 开发人员所需要的重要武器，它提供了从内容中分离应用样式和设计的机制。虽然 CSS 在 Ajax 应用中扮演至关重要的角色，但它也是构建跨浏览器应用的一大阻

碍，因为不同的浏览器支持不同的 CSS 级别。

23.3　Ajax 开发需要注意的几个问题

视频讲解：光盘\TM\lx\23\Ajax 开发需要注意的几个问题.exe

Ajax 在开发过程中需要注意以下几个问题：

1．浏览器兼容性问题

Ajax 使用了大量的 JavaScript 和 Ajax 引擎，而这些内容需要浏览器提供足够的支持。目前提供这些支持的浏览器有 IE 5.0 及以上版本、Mozilla 1.0、Netscape 7 及以上版本。Mozilla 虽然也支持 Ajax，但是提供 XMLHttpRequest 对象的方式不一样，所以使用 Ajax 程序必须测试针对各个浏览器的兼容性。

2．XMLHttpRequest 对象封装

Ajax 技术的实现主要依赖于 XMLHttpRequest 对象，但在调用其进行异步数据传输时，由于 XMLHttpRequest 对象的实例在处理事件完成后就会被销毁，所以如果不对该对象进行封装处理，在下次需要调用它时就要重新构建，而且每次调用都需要写一大段的代码，使用起来很不方便。现在很多开源的 Ajax 框架都提供了对 XMLHttpRequest 对象的封装方案，其详细内容这里不作介绍，请参考相关资料。

3．性能问题

由于 Ajax 将大量的计算从服务器端移到了客户端，这就意味着浏览器将承受更大的负担，而不再是只负责简单的文档显示。由于 Ajax 的核心语言是 JavaScript，而 JavaScript 并不以高性能知名，另外，JavaScript 对象也不是轻量级的，特别是 DOM 元素耗费了大量的内存。因此，如何提高 JavaScript 代码的性能对于 Ajax 开发者来说尤为重要。下面介绍 3 种优化 Ajax 应用执行速度的方法。

☑　优化 for 循环。

☑　将 DOM 节点附加到文档上。

☑　尽量减少点“.”号操作符的使用。

4．中文编码问题

Ajax 不支持多种字符集，它默认的字符集是 UTF-8，所以在应用 Ajax 技术的程序中应及时进行编码转换，否则程序中出现的中文字符将变成乱码。一般情况下，以下两种情况将产生中文乱码。

☑　PHP 发送中文、Ajax 接收

只需在 PHP 顶部添加如下语句：

```
header('Content-type: text/html;charset=GB2312');                    //指定发送数据的编码格式
```

XMLHttpRequest 会正确解析其中的中文。

☑ Ajax 发送中文、PHP 接收

这个比较复杂，在 Ajax 中先用 encodeURIComponent 对要提交的中文进行编码。在 PHP 页添加如下代码：

```
$GB2312string=iconv( 'UTF-8', 'gb2312//IGNORE' , $RequestAjaxString);
```

PHP 选择 MySQL 数据库时，使用如下语句设置数据库的编码类型：

```
mysql_query("set names gb2312");
```

23.4 在 PHP 中应用 Ajax 技术的典型应用

视频讲解：光盘\TM\lx\23\在 PHP 中应用 Ajax 技术的典型应用.exe

23.4.1 在 PHP 中应用 Ajax 技术检测用户名

【例 23.1】 本实例主要通过 Ajax 技术实现不刷新页面检测用户名是否被占用。（**实例位置：光盘\TM\sl\23\1**）

程序的开发步骤如下：

（1）搭建 Ajax 开发框架，代码如下：

```
<script language="javascript">
var http_request = false;
function createRequest(url) {                          //初始化对象并发出 XMLHttpRequest 请求

    http_request = false;
    if (window.XMLHttpRequest) { // Mozilla……
        http_request = new XMLHttpRequest();
        if (http_request.overrideMimeType) {
            http_request.overrideMimeType("text/xml");
        }
    } else if (window.ActiveXObject) {                 //IE 浏览器
        try {
            http_request = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
            try {
                http_request = new ActiveXObject("Microsoft.XMLHTTP");
          } catch (e) {}
        }
    }
    if (!http_request) {
        alert("不能创建 XMLHTTP 实例!");
        return false;
```



```
        }
        http_request.onreadystatechange = alertContents;    //指定响应方法
        //发出 HTTP 请求
        http_request.open("GET", url, true);
        http_request.send(null);
}
function alertContents() {                                  //处理服务器返回的信息
        if (http_request.readyState == 4) {
                if (http_request.status == 200) {
                        alert(http_request.responseText);
                } else {
                        alert('您请求的页面发现错误');
                }
        }
}
</script>
```

（2）编写 JavaScript 的自定义函数 checkName()，用于检测用户名是否为空，当用户名不为空时，调用 createRequest()函数发送请求检测用户名是否存在，代码如下：

```
<script language="javascript">
function checkName() {
        var username = form1.username.value;
        if(username=="") {
                window.alert("请添写用户名!");
                form1.username.focus();
                return false;
        }
        else {
                createRequest('checkname.php?username='+username+'&nocache='+new Date().getTime());
        }
}
</script>
```

在上面的代码中，必须添加清除缓存的代码（加粗的代码部分），否则程序将不能正确检测用户名是否被占用。

（3）在页面的适当位置添加“检测用户名”超链接，在该超链接的 onclick 事件中调用 checkName()方法弹出显示检测结果的对话框，关键代码如下：

```
<a href="#" onClick="checkName();">[检测用户名]</a>
```

（4）编写检测用户名是否唯一的 PHP 处理页 checkname.php，在该页面中使用 PHP 的 echo 语句输出检测结果，完整代码如下：

```
<?php
        $link=mysql_connect("localhost","root","root");
        mysql_select_db("db_database23",$link);
        //Ajax 中先用 encodeURIComponent 对要提交的中文进行编码
        $GB2312string=iconv( 'UTF-8', 'gb2312//IGNORE' , $RequestAjaxString);
```

```
	mysql_query("set names gb2312");
	$username=$_GET[username];
	$sql=mysql_query("select * from tb_user where name='".$username."'");
	$info=mysql_fetch_array($sql);
	header('Content-type: text/html;charset=GB2312');          //指定发送数据的编码格式为 GB2312
	if ($info){
		echo "很报歉!用户名[".$username."]已经被注册!";
	}else{
		echo "祝贺您!用户名[".$username."]没有被注册!";
	}
?>
```

运行本实例，在“用户名”文本框中输入“纯净水”，单击“检测用户名”超链接，即可在不刷新页面的情况下弹出“祝贺您!用户名[纯净水]没有被注册!”的提示对话框，如图 23.3 所示。



图 23.3　检测用户名

23.4.2　在 PHP 中应用 Ajax 技术实现博客文章类别添加

【例 23.2】 本实例主要通过 Ajax 技术实现无刷新的博客文章类别添加。（**实例位置：光盘\TM\sl\23\2**）

程序的开发步骤如下：

（1）搭建 Ajax 开发框架，具体代码如下：

```
<script language="javascript">
var http_request = false;
function createRequest(url) {
	//初始化对象并发出 XMLHttpRequest 请求
	http_request = false;
	if (window.XMLHttpRequest) {                                    //Mozilla 等其他浏览器
		http_request = new XMLHttpRequest();
		if (http_request.overrideMimeType) {
			http_request.overrideMimeType("text/xml");
		}
```



```
    } else if (window.ActiveXObject) {                                    //IE 浏览器
        try {
            http_request = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
            try {
                http_request = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e) {}
        }
    }
    if (!http_request) {
        alert("不能创建 XMLHTTP 实例!");
        return false;
    }
    http_request.onreadystatechange = alertContents;                      //指定响应方法

    http_request.open("GET", url, true);                                   //发出 HTTP 请求
    http_request.send(null);
}
function alertContents() {                                                 //处理服务器返回的信息
    if (http_request.readyState == 4) {
        if (http_request.status == 200) {
            sort_id.innerHTML=http_request.responseText;      //设置 sort_id HTML 文本替换的元素内容
        } else {
            alert('您请求的页面发现错误');
        }
    }
}
</script>
```

在上面的代码中，要特别注意的是加粗部分的代码，sort_id 是显示文章分类信息的单元格 id 属性，将在本实例的步骤（4）中介绍。innerHTML 属性声明了元素含有的 HTML 文本，不包括元素本身的开始标记和结束标记，该属性用于指定 HTML 文本替换元素的内容。

（2）编写 JavaScript 的自定义函数 check_sort()用于检测欲添加的类别名称是否为空，当类别名称文本框不为空时调用 createRequest()函数发送请求获取添加类别信息到数据库中，代码如下：

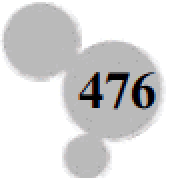
```
<script language="javascript">
function checksort() {
    var txt_sort = form1.txt_sort.value;
    if(txt_sort=="") {
        window.alert("请填写文章类别!");                    //如果文章类别文本框内容为空，弹出提示
        form1.txt_sort.focus();
        return false;
    }
    else {
        createRequest('checksort.php?txt_sort='+txt_sort);   //提交分类信息到数据处理页
    }
}
</script>
```

（3）在下拉列表中动态输出博客文章的类别信息，这里更重要的是将第一行代码中单元格的 id 属性设置为 sort_id，便于在 JavaScript 脚本中调用。另外，在“添加分类”图像的 onclick 事件中调用 checksort()方法，代码如下：

```
<td width="14%" valign="baseline" id="sort_id">
  <table border="0" cellpadding="0" cellspacing="0">
    <tr>
      <td>
        <select name="select" >
        <?php
        $link=mysql_connect("localhost","root","root");              //连接 MySQL 数据库服务器
        mysql_select_db("db_database23",$link);                      //选择数据库文件
        //Ajax 中先用 encodeURIComponent 对要提交的中文进行编码
        $GB2312string=iconv( 'UTF-8', 'gb2312//IGNORE' , $RequestAjaxString);
        mysql_query("set names gb2312");                             //设置数据库编码类型为 GB2312
        $sql=mysql_query("select distinct * from tb_sort group by sort");
        $result=mysql_fetch_object($sql);                            //检索数据表中的信息
        do{
            header('Content-type: text/html;charset=GB2312');         //指定发送数据的编码格式为 GB2312
        ?>
        <option value="<?php echo $result->sort;?>" selected><?php echo $result->sort;?></option>
        <?php
        }while($result=mysql_fetch_object($sql));
        ?>
      </select>
     </td>
     <td width="20%" height="21" align="right" valign="baseline">
     <input name="txt_sort" type="text" id="txt_sort" size="12" style="border:1px #64284A solid; height:21">
     </td>
     <td width="49%" height="21" align="left" valign="baseline">
     <img src="images/add.gif" width="67" height="23" onclick="checksort();">
     </td>
    </tr>
  </table>
</td>
```

（4）编写添加分类信息到 PHP 处理页 checksort.php，在该页面中首先从数据表中获取博客分类信息，然后添加到数据库中，最后显示在下拉列表中，完整代码如下：

```
<?php
     $link=mysql_connect("localhost","root","root");
     mysql_select_db("db_database23",$link);
     //Ajax 中先用 encodeURIComponent 对要提交的中文进行编码
     $GB2312string=iconv( 'UTF-8', 'gb2312//IGNORE',$RequestAjaxString);
     mysql_query("set names gb2312");
     $sort=$_GET[txt_sort];
     mysql_query("insert into tb_sort(sort) values('$sort')");
     header('Content-type: text/html;charset=GB2312');              //指定发送数据的编码格式为 GB2312
?>
```



```
<!-- 下面的代码部分是单元格 id 属性中的代码部分，与步骤（3）等同，只是不包括元素本身的开始标记和结束
标记<td width="14%" valign="baseline" id="sort_id">，该属性用于指定 HTML 文本替换元素的内容。 --!>
<table border="0" cellpadding="0" cellspacing="0">
  <tr>
     <td>
     <select name="select" >
     <?php
          $link=mysql_connect("localhost","root","root");                //连接 MySQL 数据库服务器
          mysql_select_db("db_database23",$link);                        //选择数据库文件
          //Ajax 中先用 encodeURIComponent 对要提交的中文进行编码
          $GB2312string=iconv( 'UTF-8', 'gb2312//IGNORE' , $RequestAjaxString);
          mysql_query("set names gb2312");                               //设置数据库编码类型为 GB2312
          $sql=mysql_query("select distinct * from tb_sort group by sort");
          $result=mysql_fetch_object($sql);                              //检索数据表中的信息
          do{
               header('Content-type: text/html;charset=GB2312');    //指定发送数据的编码格式为 GB2312
     ?>
     <option value="<?php echo $result->sort;?>" selected><?php echo $result->sort;?></option>
     <?php
          }while($result=mysql_fetch_object($sql));
     ?>
    </select>
    </td>
    <td width="20%" height="21" align="right" valign="baseline">
    <input name="txt_sort" type="text" id="txt_sort" size="12" style="border:1px #64284A solid; height:21">
    </td>
    <td width="49%" height="21" align="left" valign="baseline">
    <img src="images/add.gif" width="67" height="23" onclick="checksort();">
    </td>
  </tr>
</table>
```

运行本实例，在“文章类别”后面的文本框中输入“心灵感悟”，单击“添加分类”按钮，即可在“文章类别”下拉列表框中成功添加该分类信息，如图 23.4 所示。



图 23.4　在 PHP 中应用 Ajax 技术实现博客文章类别添加

23.5 小　　结

本章主要介绍了应用 PHP 开发动态网站时的一些高级技术，读者应该认真学习并掌握。通过这些技术可以使编程水平上升到一个新的层次。例如，使用 Ajax 技术可以实现很多无刷新效果，增强页面的友好感。

23.6 练习与实践

1．应用 Ajax 技术实现无刷新的级联下拉列表。在“所属大类”下拉列表框中选择“家居日用”列表项后，在“所属小类”下拉列表框中将显示属于该类别下的全部子类。**（答案位置：光盘\TM\sl\23\3）**

2．开发一个带记忆功能的查询功能，系统自动记录每次用户输入的查询关键字，并且当用户再次输入时，只要输入已经记录的关键字的第一个字符，系统就会自动查询出与该关键字相匹配的全部选项，并显示在列表中供用户选择。**（答案位置：光盘\TM\sl\23\4）**

第 4 篇

项目实战

本篇通过 Smarty 模板技术、ADODB 类库、Ajax 等主流技术实现一个大型、完整的电子商务平台，运用软件工程的设计思想，让读者学习如何进行网站项目的实践开发。书中按照编写项目计划书→系统设计→数据库设计→创建项目→实现项目→运行项目→解决开发常见问题→发布网站的过程进行介绍，带领读者一步步亲身体验开发项目的全过程。

第24章

应用 Smarty 模板开发电子商务网站

（视频讲解：2 小时 5 分钟）

随着 20 世纪 PC（个人计算机）的发展和互联网的普及，电子商务从报文时代进入到了 Internet 时代，并逐渐被大众所了解和接受。电子商务（Electronic Commerce，EC）是目前发展较快的一种商务模式。迄今为止，不同领域的人对 EC 的理解各有不同。简单地说，EC 是一种基于 Internet，利用计算机硬件、软件等现有设备和协议进行各种商务活动的方式。

通过阅读本章，您可以：

- 了解如何进行系统分析
- 掌握编写项目计划书的方法
- 了解数据库设计流程
- 熟悉搭建系统架构的方法
- 掌握注册即时验证的实现方法
- 掌握简单的树形菜单的实现方法
- 掌握购物车的实现方法
- 掌握订单的处理方法
- 了解如何注册域名和虚拟空间
- 掌握发布网站的方法

24.1 系 统 分 析

视频讲解：光盘\TM\lx\24\系统分析.exe

24.1.1 需求分析

随着“地球村”概念的兴起，网络已经深入到人们生活的每一个角落。世界越来越小，信息的传播越来越快，内容也越来越丰富。现在，人们对于在网络上寻求信息和服务已不再满足于简单的信息获取上，更多的是需要在网上实现方便的、便捷的、可交互式的网络服务。电子商务正好满足了人们的需求。它可以让人们在网上实现互动的交流及足不出户地购买产品，向企业发表自己的意见、服务需求及有关投诉，还可以通过网站的交互式操作向企业进行产品咨询、获取回馈及技术支持。精明的商家绝不会错过这样庞大的市场，越来越多的企业参与到电子商务活动中。加入电子商务的行列也许不会让企业马上见到效益，但不加入则一定会被时代所抛弃。

24.1.2 编写项目计划书

项目计划书是项目负责人为投资者编写的一份详细的开发报告，包括开发背景、开发原因、可行性分析、成本预算和开发周期等。给投资者的计划书无论详细还是简短，都要以引起投资者对项目的兴趣为目的，在真实的基础上，清楚明确地表达该项目对投资人的必要性和迫切性。本节先来介绍一个典型的电子商务项目开发计划书。

电子商务项目开发计划书

编号：0724

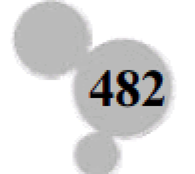
版本：v1.0

拟制人：邹甜甜

审核人：高盈月

批准人：赛凯臣

批准日期：二零零九年八月二日



电子商务平台项目计划书

一、项目开发背景

自 20 世纪 90 年代，互联网的蓬勃发展，为企业提供了一个全新的机遇。企业网站、电子商务成为热门话题。其中，电子商务更是关系到经济结构、产业升级和国家整体经济竞争力。为此，我国已经将发展电子商务列为信息化建设的重要内容，并努力创造条件，积极地推进电子商务的发展。

据美国在线（AOL）和 Henley Centre 联合进行的一项调查显示：国外有 80%的受调查者会选择网上购物或寻求帮助，10%的受调查者会选择熟悉的品牌或厂商来购买。而在国内，自 1997 年拉开了电子商务的序幕，到 2007 年短短的 10 年时间里，全国已有 4 万家商业网站，几乎每天都有新的网站诞生，厂商所在地也从上海、广州、深圳等沿海发达地区扩展到全国各大中城市。

二、电子商务的概念及特点

电子商务（Electronic Commerce），简称 EC，是目前发展较快的一种商务模式。不同领域的人群对 EC 的理解各有不同，综合来说，电子商务是通过互联网进行的商务活动，它已经渗透了各个领域，如服务、金融、销售等。

目前来说，电子商务主要有两种模式。一种是纯粹的电子商务网站，如阿里巴巴，它为交易双方提供了一个丰富的信息平台；另一种是传统产业的电子商务模式，它借助互联网进行特定商品的销售及采购。

三、电子商务的优势

电子商务的优势是显而易见的，主要有以下几个方面：

- 大大节约了企业的成本预算，如人工费、场地费等。
- 降低了产品的零售价格。因为是和消费者直接交流沟通，取消了代理、批发、商场等费用，在同行业的竞争中占据优势。
- 扩展了业务范围，只要能够上网的人，就有机会成为客户，一定程度上消除了地域的影响。
- 加速了资本流通。
- 实现 24 小时连续服务。

四、可行性分析

- 市场前景。数据表明：2007 年中国 B2B 电子商务交易规模为 12400 亿元，比 2006 年增长 24.5%。预计未来两年我国 B2B 电子商务交易规模将继续高速增长，2008 年已经达到 16200 亿元，2009 年交易规模已经达 21300 亿元。B2C 网站总收入为 52.2 亿元，2009 年已经达到 98.6 亿元。而且，随着网络购物环境的好转，未来两年 B2C 电子商务交易模式将更受欢迎，用户数和年平均消费金额均会提高。这里所蕴含的商机毋庸置疑。
- 技术可行性。电子商务相关的技术（如认证加密技术、数据备份技术、网上交易技术）已经十分完善，能够保障系统正常、无错地运行。
- 经济可行性。通过网络进行商务活动，可以足不出户地完成交易。既可节省交通费，又减少了务工人员费用，同时，前期可以使用小型服务器或虚拟空间以减少成本预算。

五、项目实施的技术方案

- 为了节约成本，使用 PHP 做开发，数据库使用 MySQL。
- 采用 Smarty 模板，可以最大限度地加快网页访问速度。
- 考虑到以后可能升级数据库，使用 ADODB 类库连接。
- 使用 Ajax 技术来提高交互性。
- 提供多种付款方式，包括网银、邮购、汇款等。
- 提供在线咨询。
- 操作简单容易，只需要简单指点便可使用，无须专业人员。

第 2 页共 4 页

电子商务平台项目计划书

六、总体投入

报价

顶级域名	.com 或.net 结尾（英文域名）	￥100/年
虚拟主机	智尊 U300（240MB/赠送 124MB　邮局/50MB mysql/php/perl）	￥750/年
电子商务平台	电子商务网上交易系统	￥20000/套（包括一年的免费维护，如需二次开发费用另算）
留言板	用户提交留言，不需审核立即显示，管理员可随时回复删除相关主题，是一套人性化的沟通工具，有助于搞好客主关系	￥500/个
友情链接	管理员可随时添加相关链接	￥500/个
公告版		￥500/个
LOGO		￥200/个
搜索引擎优化	使网站能够被搜索引擎抓取，并排名靠前，利于网站的推广	赠送

共计：域名+虚拟主机+电子商务系统+留言板+友情链接+公告版=¥22450，以后每年只需交¥850（域名+主机）即可。

七、人员安排

人员安排

项目负责人	1 名（邹甜甜）	负责项目的总体设计以及与企业沟通
系统策划	1 名（刘明松）	负责网站的策划和制作
网站开发人员	4 名（潘明远、孙海、刘忻南、梁淼）	负责系统后台开发
美工	1 名（董明）	负责页面和 LOGO 设计
测试人员	1 名（刘杏）	负责系统的后期调试

八、后期维护

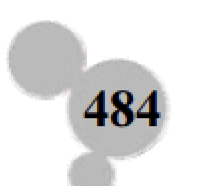
在系统正式使用的一年时间内，将对系统进行免费维护（添加系统功能或修改框架除外），及时修改程序缺陷，还为客户提供技术培训。同时，保证运行速度和系统安全。

九、系统功能模块

本系统分前台和后台两部分。

- 前台包括
  - 登录及注册模块：包括会员的注册、登录和找回密码功能。
  - 商品模块：包括商品的分类浏览和查看。
  - 商品搜索模块：包括精确查询和模糊查询两种模式。实现对商品任意信息的搜索。
  - 购物车模块：帮助会员完成购物功能。
  - 在线支付模块：支持工行、招行等各个银行的信用卡、支付卡。
  - 公告栏模块：最新的咨询、活动、声明等。
  - 友情链接模块：行业内网站之间的互换链接栏目的要求，实现网站的免费推广与宣传。
  - 留言模块：包括留言板、在线 QQ 等客户和企业的交流平台。
- 后台包括
  - 商品管理：包括对商品的增删改查等操作和商品的类别管理。
  - 会员管理模块：包括对会员的信息和留言管理。
  - 订单管理模块：管理员可以对客户提交的订单进行编辑、查看和处理。
  - 公告管理模块：对公告版的信息进行添加、删除、修改等操作。

第 3 页共 4 页



电子商务平台项目计划书

十、项目安排及开发时间表

项目共分 3 部分进行。

第一部分：页面验收。从签订合同之日起，美工开始设计效果图，然后交给客户验收。如果客户不满意，则美工重新设计或修改。只有客户满意后才继续进行。

第二部分：客户验收合格后，网站开发人员开始实现程序的功能模块。如果在开发过程中发现漏洞或是其他一些技术原因，将尽快和用户协调商定，以免造成严重影响。

第三部分：模块测试。系统开发完成后，由测试人员进行调试，将错误降到最低。

项目开始日期：2009 年 11 月 20 日

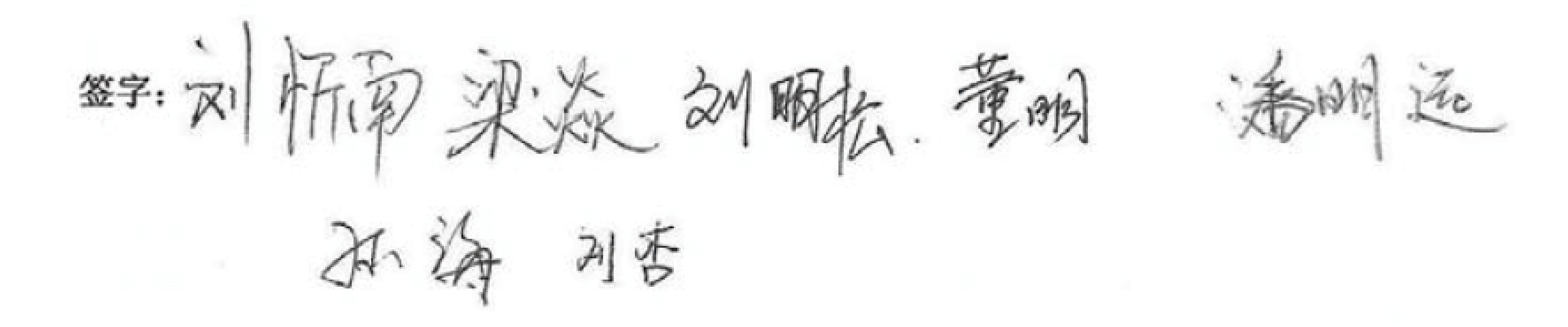
项目完成日期：2009 年 12 月 24 日

开发时间表

任务安排	开始时间	完成时间	工作日	负责人
验收页面设计	2009-12-03	2009-12-09	7	董明
系统策划分析	2009-12-03	2009-12-05	3	刘明松
商品模块设计	2009-12-10	2009-12-12	3	潘明远
商品搜索	2009-12-13	2009-12-15	3	
商品管理模块	2009-12-16	2009-12-18	3	
登录及注册模块设计	2009-12-10	2009-12-12	3	孙海
留言模块	2009-12-13	2009-12-15	3	
会员管理模块	2009-12-17	2009-12-18	3	
在线支付模块	2009-12-10	2009-12-12	3	刘忻南
购物车模块	2009-12-13	2009-12-15	3	
订单管理模块	2009-12-16	2009-12-18	3	
数据库设计	2009-12-06	2009-12-08	3	梁焱
公告栏模块	2009-12-10	2009-12-11	2	
友情链接模块	2009-12-12	2009-12-13	2	
公告管理模块	2009-12-14	2009-12-16	3	
系统测试	2009-12-19	2009-12-24	5	刘杏

十一、责任人签字

所有参与开发的人员，在开发期间，要严格遵守保密协议，凡是涉及客户企业的商业机密、内部资料等，均不得外泄，否则将追究法律责任。



第 4 页共 4 页

24.2　系统设计

视频讲解：光盘\TM\lx\24\系统设计.exe

24.2.1　系统目标

根据客户提供的需求和对实际情况的考察与分析，该电子商务网站应该具备如下特点：

- ☑ 首页设计要能够吸引用户的目光，整个页面要以简洁为主，突出重点。
- ☑ 可操作性强，避免复杂的、有异议的链接。
- ☑ 浏览速度快，尽量避免长时间打不开页面的情况发生。
- ☑ 商品信息部分有实物图例，图像清楚、文字醒目。
- ☑ 详细的商品查询功能，可以通过商品的各个属性来搜索。
- ☑ 详细的流程介绍，从浏览商品到购买结账，各个步骤之间的联系最好能以图例来说明。
- ☑ 提供在线咨询。
- ☑ 后台可以对用户信息和商品信息进行详尽查看和管理。
- ☑ 订单管理。
- ☑ 易维护，并提供二次开发支持。

24.2.2　系统功能结构

电子商务平台分前台系统和后台系统。下面分别给出前、后台的系统功能结构图。电子商务前台系统功能结构图如图 24.1 所示。

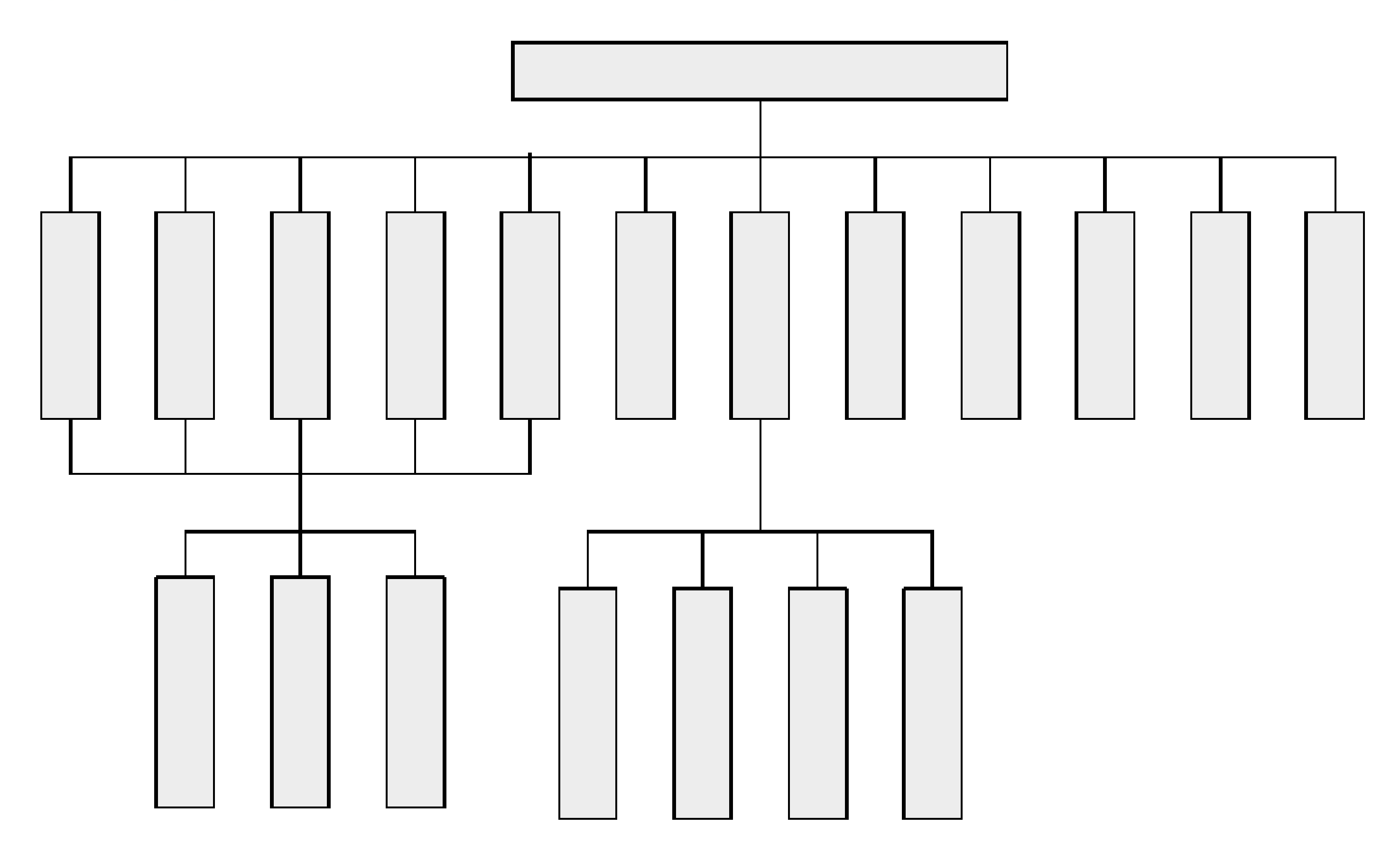


图 24.1　电子商务前台系统功能结构图

电子商务后台系统功能结构图如图 24.2 所示。



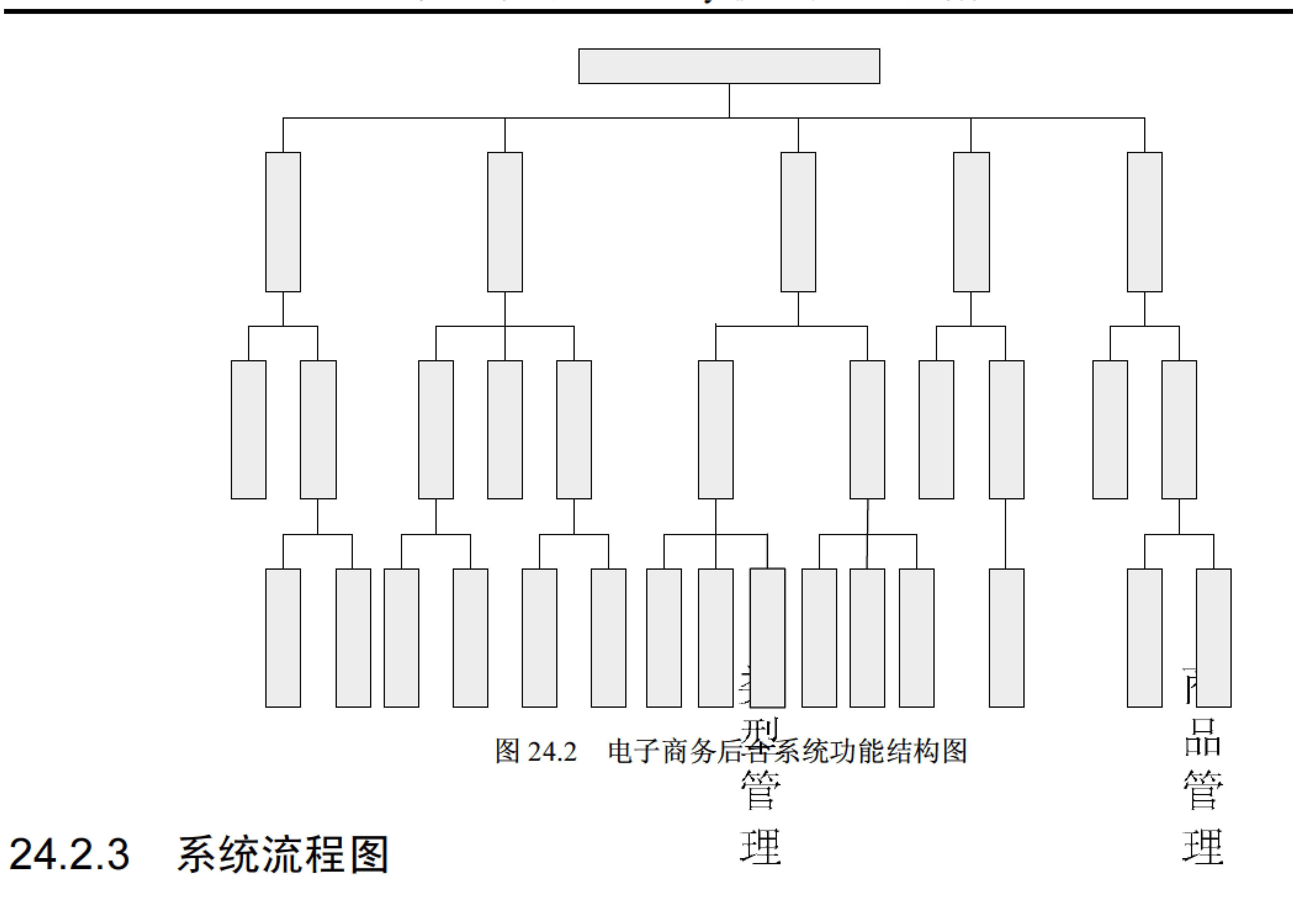


图 24.2　电子商务后台系统功能结构图

24.2.3　系统流程图

为了便于开发人员了解系统各个功能模块之间的联系及完整的购物流程，下面给出了系统的流程图，如图 24.3 所示。

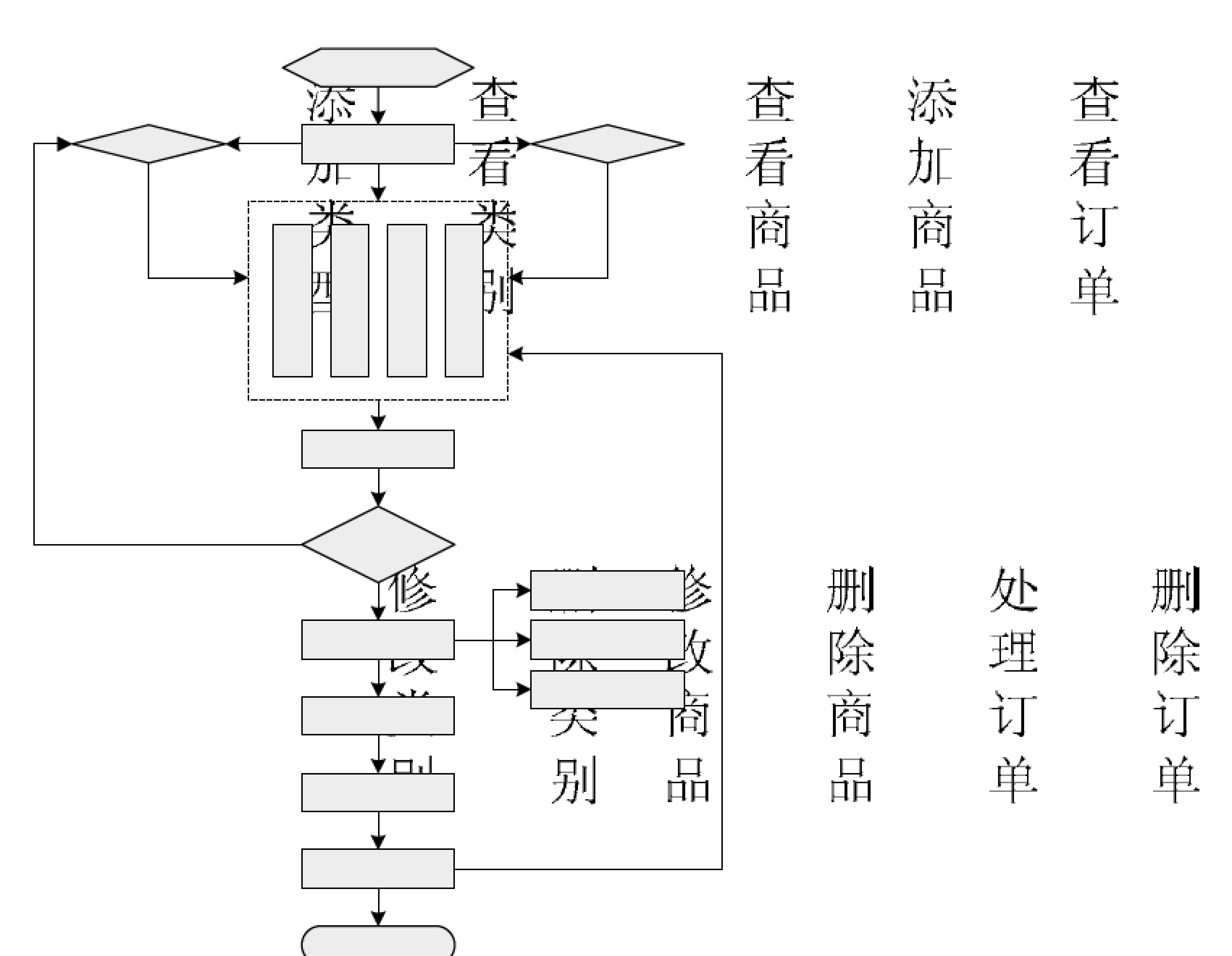


图 24.3　系统流程图

24.3 软件开发环境

视频讲解：光盘\TM\lx\24\软件开发环境.exe

在开发电子商务平台时，该项目使用的软件开发环境如下：

1. 服务器端

- ☑ 操作系统：Windows 2003 Server/Linux（推荐）。
- ☑ 服务器：Apache 2.2.8。
- ☑ PHP 软件：PHP 5.2.6。
- ☑ 数据库：MySQL 5.0.51。
- ☑ MySQL 图形化管理软件：phpMyAdmin-2.10.3。
- ☑ 开发工具：Dreamweaver 8。
- ☑ 浏览器：IE 6.0 及以上版本。
- ☑ 分辨率：最佳效果为 1024×768 像素。

2. 客户端

- ☑ 浏览器：推荐 IE 6.0 及以上版本。
- ☑ 分辨率：最佳效果为 1024×768 像素。

24.4 数据库与数据表的设计

视频讲解：光盘\TM\lx\24\数据库与数据表的设计.exe

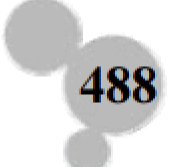
无论是什么系统软件，其最根本的功能就是对数据的操作与使用。所以，一定要先做好数据的分析、设计与实现，然后才实现对应的功能模块。

24.4.1 数据库分析

根据需求分析和系统的功能流程图，找出需要保存的信息数据（也可以理解为现实世界中的实体），并将其转化为原始数据（属性类型）形式。这种描述现实世界的概念模型，可以使用 E-R 图来表示。也就是实体-联系图。最后将 E-R 图转换为关系数据库。这里重点介绍几个 E-R 图。

1. 会员信息实体

会员信息实体包括编号、用户名、密码、E-mail、身份证号、联系电话、QQ 号、密码提示、密码答案、邮编、注册时间、真实姓名等属性。会员信息实体 E-R 图如图 24.4 所示。



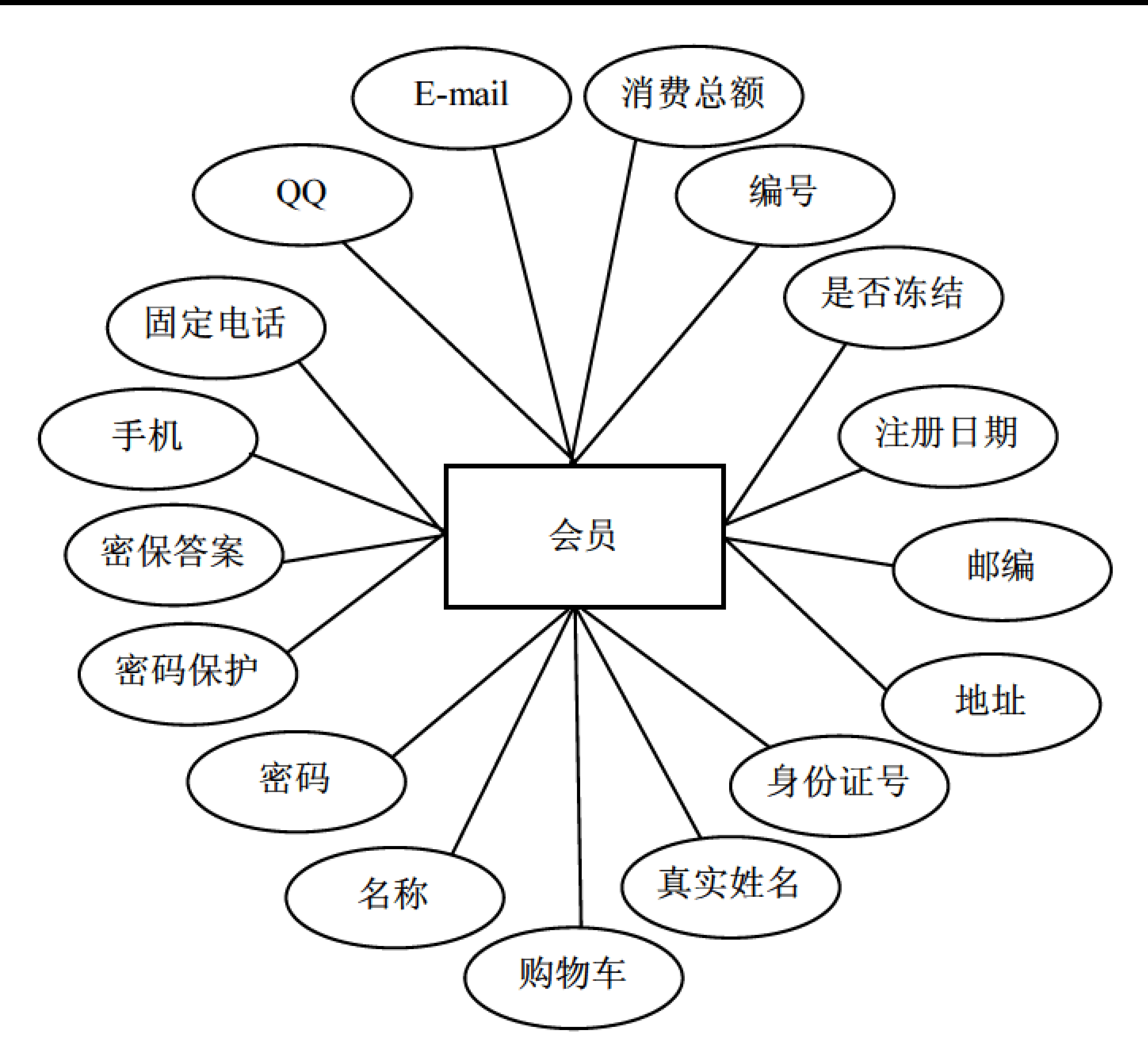


图 24.4　会员信息实体 E-R 图

2．商品信息实体

商品信息实体包括编号、名称、上市时间、添加时间、型号、图片、库存、销售、商品类型、会员价、市场价、是否打折等属性。商品信息实体 E-R 图如图 24.5 所示。

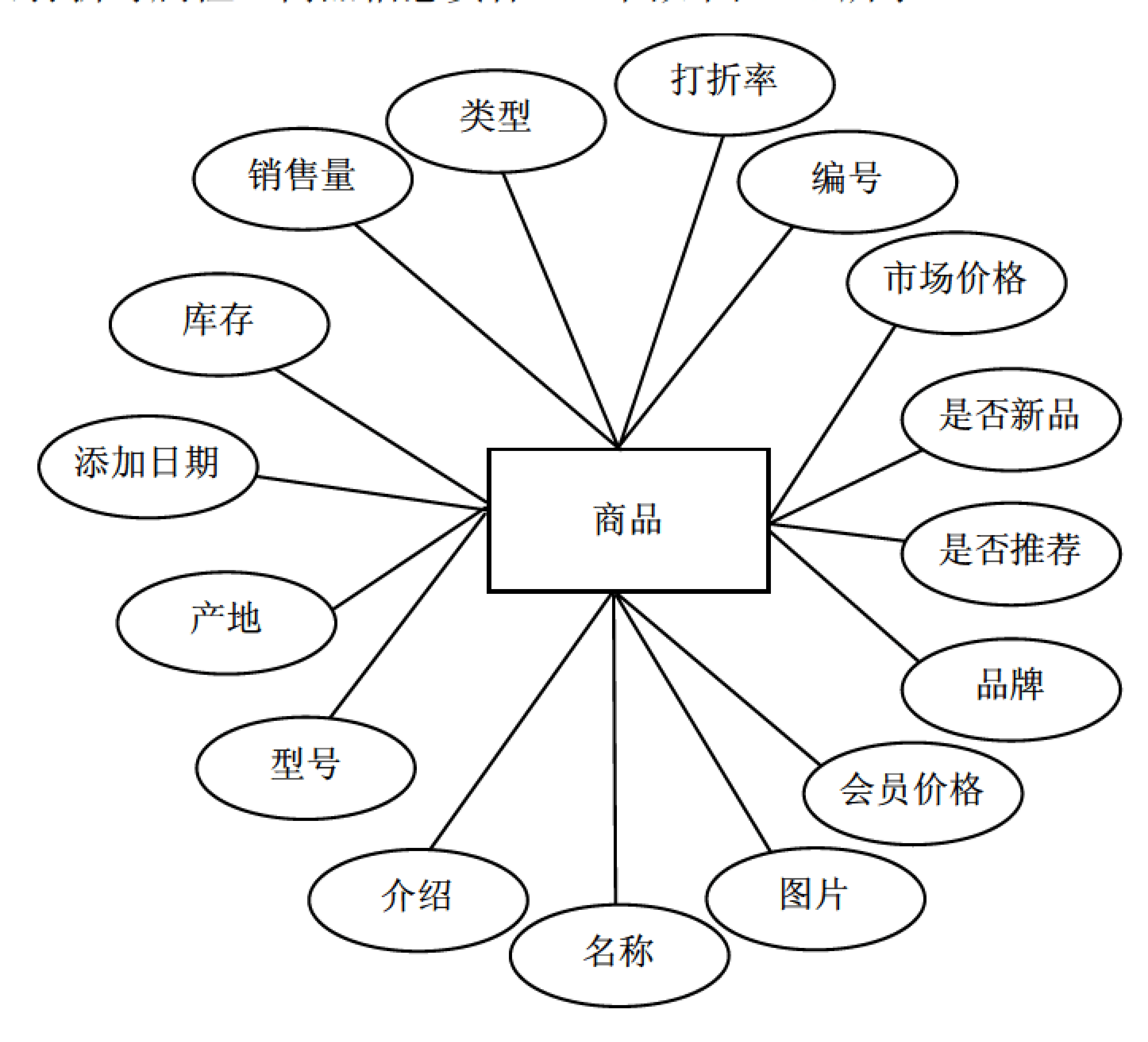


图 24.5　商品信息实体 E-R 图

3．商品订单实体

商品订单实体包括编号、订单号、商品 id、数量、单价、打折率、收货人姓名、收货地址、邮编、联系电话、E-mail、收货方式、支付方式、订单时间、发货人姓名、状态、价格总计等属性。商品订单实体 E-R 图如图 24.6 所示。

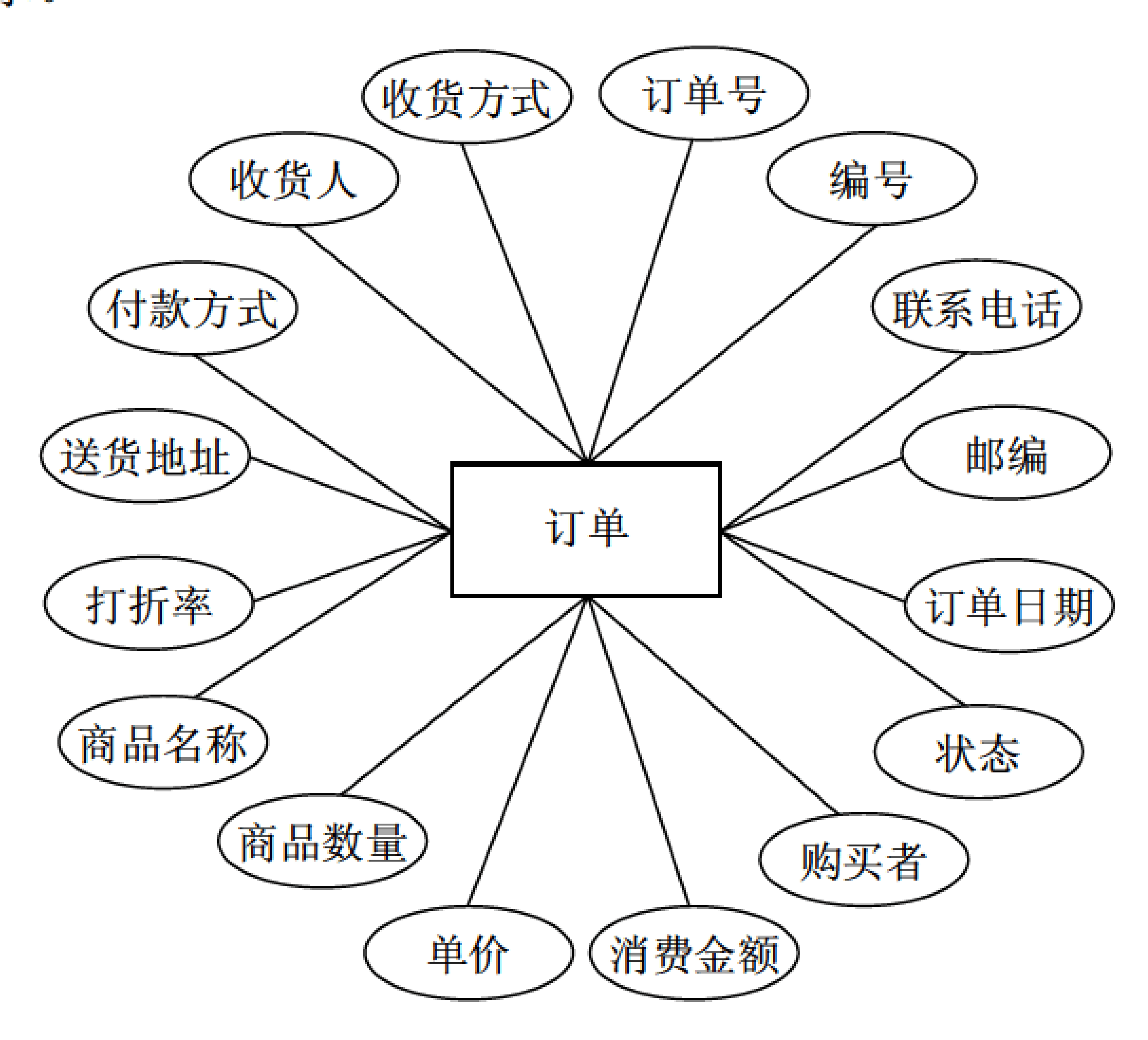


图 24.6　商品订单实体 E-R 图

4．商品评价实体

商品评价实体包括编号、用户编号、商品编号、评价内容、评价时间等属性。商品评价实体 E-R 图如图 24.7 所示。

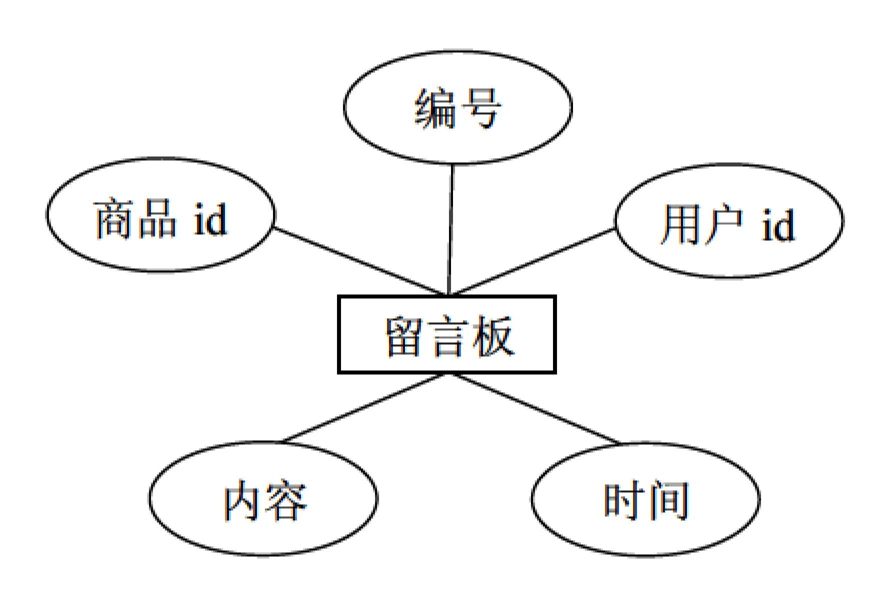
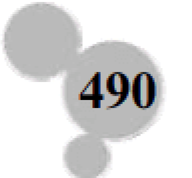


图 24.7　商品评价实体 E-R 图

除了上面介绍的 4 个 E-R 图之外，还有公告实体、管理员实体、类型实体和友情链接实体等，限于篇幅，这里仅列出主要的实体 E-R 图。

24.4.2　创建数据库和数据表

系统 E-R 图设计完成后，接下来根据 E-R 图来创建数据库和数据表。关于数据库和数据表的创建



可参考第 16 章数据库基础，这里不再赘述。首先来看一下电子商务平台所使用的数据表情况，如图 24.8 所示。

服务器: localhost ▸ 数据库: db_shop

表	类型	整理	说明
tb_admin	MyISAM	gb2312_chinese_ci	管理员信息表
tb_class	MyISAM	gb2312_chinese_ci	商品类型表
tb_commo	MyISAM	gb2312_chinese_ci	商品信息表
tb_form	MyISAM	gb2312_chinese_ci	商品订单表
tb_links	MyISAM	gb2312_chinese_ci	友情链接表
tb_opinion	MyISAM	gb2312_chinese_ci	商品评论表
tb_public	MyISAM	gb2312_chinese_ci	公告信息表
tb_user	MyISAM	gb2312_chinese_ci	会员信息表

图 24.8　电子商务数据表

下面来看各个数据表的结构和字段说明。

1. tb_admin（管理员信息表）

管理员信息表主要用于存储管理员的信息，其结构如图 24.9 所示。

服务器: localhost ▸ 数据库: db_shop ▸ 表 : tb_admin

字段	类型	整理	属性	Null	默认	额外	说明
id	int(4)			否		auto_increment	自动编号
name	varchar(50)	gb2312_chinese_ci		否			管理员账号
pwd	varchar(20)	gb2312_chinese_ci		否			管理员密码

图 24.9　管理员信息表结构

2. tb_class（商品类型表）

商品类型列表主要用于添加商品的类别，可以设定多个子类别（目前最多只能到二级子类别），其结构如图 24.10 所示。

服务器: localhost ▸ 数据库: db_shop ▸ 表 : tb_class

字段	类型	整理	属性	Null	默认	额外	说明
id	int(4)			否		auto_increment	自动编号
name	varchar(20)	gb2312_chinese_ci		否			类型名称
level	int(1)			否	1		等级

图 24.10　商品类型表结构

3. tb_commo（商品信息表）

商品信息表主要用于存储关于商品的相关信息，其结构如图 24.11 所示。

服务器: localhost ▸ 数据库: db_shop ▸ 表 : tb_commo

字段	类型	整理	属性	Null	默认	额外	说明
id	int(4)			否		auto_increment	自动编号
name	varchar(50)	gb2312_chinese_ci		否			商品名称
pics	varchar(200)	gb2312_chinese_ci		否			商品图片
info	mediumtext	gb2312_chinese_ci		否			商品介绍
addtime	date			否			添加时间
area	varchar(50)	gb2312_chinese_ci		否			商品产地
model	varchar(50)	gb2312_chinese_ci		否			商品型号
class	varchar(50)	gb2312_chinese_ci		否			商品类别
brand	varchar(50)	gb2312_chinese_ci		否			商品品牌
stocks	int(4)			否	1		商品库存
sell	int(4)			否	0		商品销量
m_price	float			否			市场价格
v_price	float			否			会员价格
fold	int(2)			否	9		打折率
isnew	int(1)			否	1		是否新品
isnom	int(1)			否	0		是否推荐

图 24.11　商品信息表结构

4．tb_form（商品订单表）

商品订单表主要用于存储商品的订单信息，其结构如图 24.12 所示。

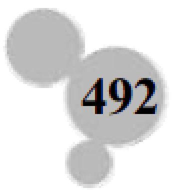
服务器: localhost ▸ 数据库: db_shop ▸ 表 : tb_form

字段	类型	整理	属性	Null	额外	说明
id	int(4)			否	auto_increment	自动编号
formid	varchar(125)	gb2312_chinese_ci		否		订单号
commo_id	varchar(100)	gb2312_chinese_ci		否		商品id
commo_name	varchar(50)	gb2312_chinese_ci		否		商品名称
commo_num	varchar(100)	gb2312_chinese_ci		否		商品数量
agoprice	varchar(50)	gb2312_chinese_ci		否		商品价格
fold	varchar(50)	gb2312_chinese_ci		否		商品折率
total	varchar(50)	gb2312_chinese_ci		否		总金额
vendee	varchar(50)	gb2312_chinese_ci		否		订单用户
taker	varchar(50)	gb2312_chinese_ci		否		收货人
address	varchar(200)	gb2312_chinese_ci		否		收货地址
tel	varchar(20)	gb2312_chinese_ci		否		移动电话
code	varchar(10)	gb2312_chinese_ci		否		邮编
pay_method	varchar(20)	gb2312_chinese_ci		否		付款方式
del_method	varchar(20)	gb2312_chinese_ci		否		送货方式
formtime	timestamp			否		订单时间
state	int(1)			否		订单状态

图 24.12　商品订单表结构

5．tb_public（公告信息表）

公告信息表主要用于展示网站的最新活动和最新消息，包括发布时间、公告标题和公告内容，其结构如图 24.13 所示。



服务器: localhost ▸ 数据库: db_shop ▸ 表 : tb_public

字段	类型	整理	属性	Null	默认	额外	说明
id	int(4)			否		auto_increment	自动编号
title	varchar(50)	gb2312_chinese_ci		否			公告标题
content	mediumtext	gb2312_chinese_ci		否			公告内容
is_time	date			否			发布时间

图 24.13　公告信息表结构

6. tb_user（会员信息表）

会员信息表主要用于存储用户的基本信息，其结构如图 24.14 所示。

服务器: localhost ▸ 数据库: db_shop ▸ 表 : tb_user

字段	类型	整理	属性	Null	默认	额外	说明
id	int(4)			否		auto_increment	自动编号
name	varchar(50)	gb2312_chinese_ci		否			会员名称
password	varchar(50)	gb2312_chinese_ci		否			密码
question	varchar(50)	gb2312_chinese_ci		否			密码保护
answer	varchar(50)	gb2312_chinese_ci		否			问题答案
consume	float			否	0		消费总额
realname	varchar(50)	gb2312_chinese_ci		否			真实姓名
card	varchar(20)	gb2312_chinese_ci		否			身份证号
tel	varchar(20)	gb2312_chinese_ci		否			移动电话
phone	varchar(20)	gb2312_chinese_ci		否			固定电话
Email	varchar(25)	gb2312_chinese_ci		否			Email
QQ	varchar(10)	gb2312_chinese_ci		否			QQ
code	varchar(10)	gb2312_chinese_ci		否			邮编
address	varchar(200)	gb2312_chinese_ci		否			地址
addtime	date			否			注册时间
isfreeze	int(1)			否	0		是否冻结
shopping	varchar(200)	gb2312_chinese_ci		否			购物车信息

图 24.14　会员信息表结构

此外还有友情链接表和商品评论表，限于篇幅，这里不再介绍，读者可参考本书附赠光盘中的数据库文件。

24.5　搭建系统框架

视频讲解：光盘\TM\lx\24\搭建系统框架.exe

编写代码之前，可以把系统中可能用到的文件夹先创建出来（例如，创建一个名为 images 的文件夹，用于保存程序中所使用的图片），这样不但可以方便以后的开发工作，也可以规范系统的整体架构。因为本项目使用的是 Smarty+ADODB 技术，所以目录较多。下面简要介绍一下本系统的目录结构（到三级目录），如图 24.15 所示。

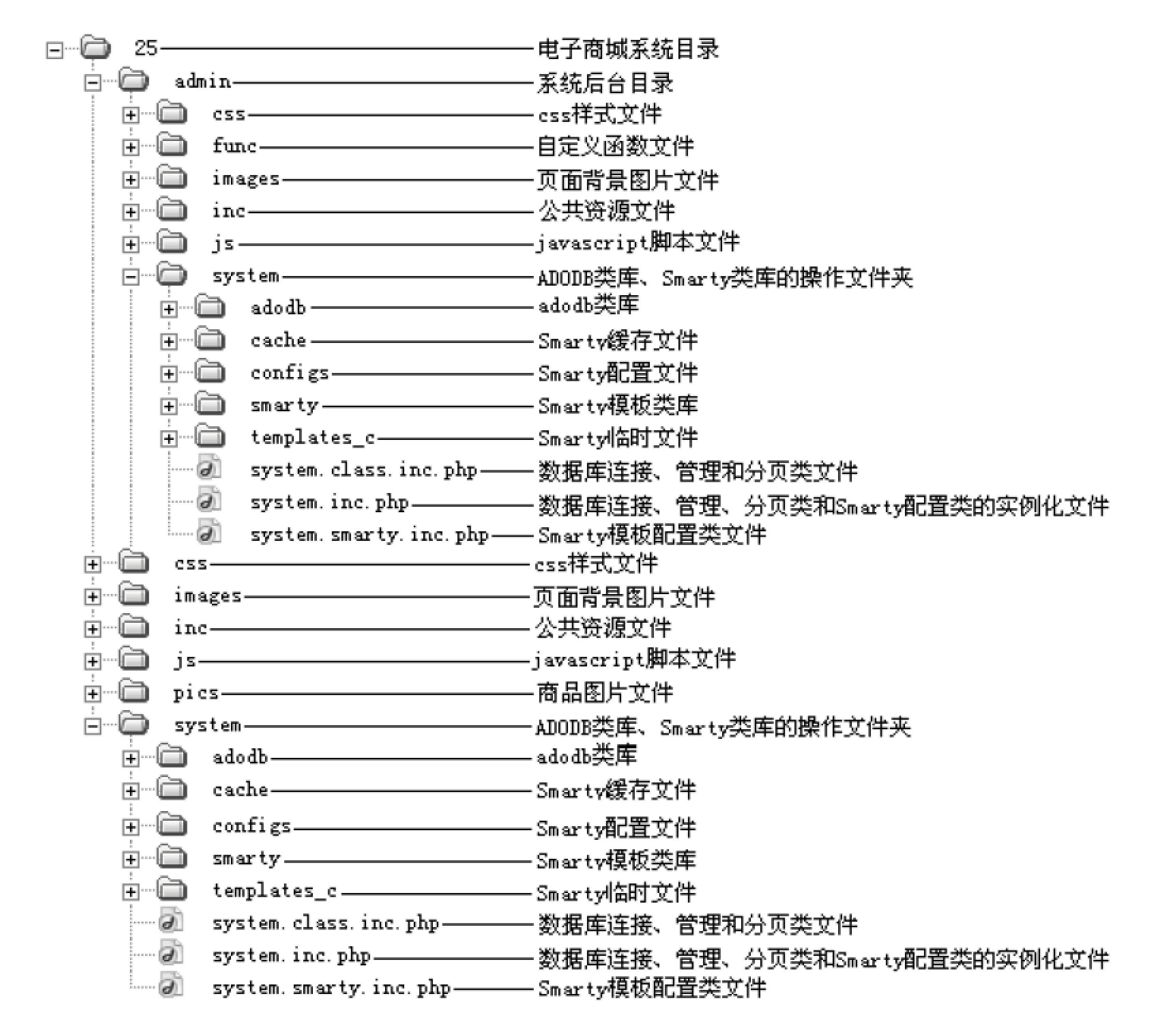


图 24.15　系统目录结构图

24.6　公共文件设计

视频讲解：光盘\TM\lx\24\公共文件设计.exe

公共模块就是将多个页面都可能使用到的代码写成单独的文件，在使用时只要用 include 或 require 语句将文件包含进来即可。如本系统中的数据库连接、管理和分页类文件，Smarty 模板配置类文件，类的实例化文件，CSS 样式表文件，js 脚本文件等。以前台系统为例，下面给出主要的公共文件，后台的公共文件与前台大同小异。

24.6.1　数据库连接、管理和分页类文件

在数据库连接、管理和分页类文件中，定义了 3 个类。分别是 ConnDB 数据库连接类，用于通过 ADODB 连接 MySQL 数据库；AdminDB 数据库管理类，使用 ADODBn 类库中的方法执行对数据库中数据的查询、添加、更新和删除操作；SepPage 分页类，用于对商城中的数据进行分页输出。

【例 24.1】 代码位置：光盘\TM\sl\24\system\system.class.inc.php

```
<?php
//数据库连接类
```



```
class ConnDB{
    var $dbtype;
    var $host;
    var $user;
    var $pwd;
    var $dbname;
    var $debug;
    var $conn;
   function ConnDB($dbtype,$host,$user,$pwd,$dbname,$debug=false){    //构造方法，为成员变量赋值
        $this->dbtype=$dbtype;
        $this->host=$host;
        $this->user=$user;
        $this->pwd=$pwd;
        $this->dbname=$dbname;
        $this->debug=$debug;
    }
   function GetConnId(){                                     //实现与不同数据库的连接并返回连接对象
    require("adodb/adodb.inc.php");                          //调用 ADODB 类库文件
        if($this->dbtype=="mysql"){                          //判断成员变量传递的数据库类型
                $this->conn=NewADOConnection("mysql");       //执行与 MySQL 数据库的连接
            $this->conn->Connect($this->host,$this->user,$this->pwd,$this->dbname);
                                                             //数据库连接的用户、密码
        }
        $this->conn->Execute("set names gb2312");            //设置数据库的编码格式
        if($this->dbtype=="mysql")
            $this->conn->debug=$this->debug;
        return $this->conn;                                  //返回连接对象
   }
    function CloseConnId(){                                  //定义关闭数据库的方法
        $this->conn->Disconnect();                           //执行关闭的操作
   }
}
//数据库管理类
class AdminDB{
    function ExecSQL($sqlstr,$conn){                    //定义方法，参数为 sql 语句和连接数据库返回的对象
        $sqltype=strtolower(substr(trim($sqlstr),0,6));     //截取 sql 中的前 6 个字符串，并转换成小写
        $rs=$conn->Execute($sqlstr);                        //执行 sql 语句
        if($sqltype=="select"){                             //判断如果 sql 语句的类型为 select
            $array=$rs->GetRows();                          //执行该语句，获取查询结果
            if(count($array)==0 || $rs==false)              //判断语句是否执行成功
                return false;                               //如果查询结果为 0，或者执行失败，则返回 false
            else
                return $array;                              //否则返回查询结果的数组
        }elseif ($sqltype=="update" || $sqltype=="insert" || $sqltype=="delete"){
            //判断如果 sql 语句类型不为 select，则执行如下操作
            if($rs)
               return true;                                 //执行成功返回 true
            else
```

```
                return false;                           //否则返回 false
            }
        }
}
//分页类
class SepPage{
        var $rs;
        var $pagesize;
        var $nowpage;
        var $nowpages;
        var $array;
        var $conn;
        var $sqlstr;
        function ShowDate($sqlstr,$conn,$pagesize,$nowpage){  //定义方法
            if(!isset($nowpage) || $nowpage=="")              //判断变量值是否为空
                $this->nowpage=10;                            //定义每页输出的记录数
            else
                $this->nowpage=$nowpage;
            $this->pagesize=$pagesize;                        //定义每页输出的记录数
            $this->conn=$conn;                                //连接数据库返回的标识
            $this->sqlstr=$sqlstr;                            //执行的查询语句
            $this->rs=$this->conn->PageExecute($this->sqlstr,$this->pagesize,$this->nowpage);
            @$this->array=$this->rs->GetRows();               //获取记录数
                if(count($this->array)==0 || $this->rs==false)
                    return false;
                else
                    return $this->array;
        }
        function ShowPage($contentname,$utits,$anothersearchstr,$class,$page){
            $allrs=$this->conn->Execute($this->sqlstr);       //执行查询语句
            $record=count($allrs->GetRows());                 //统计记录总数
            $pagecount=ceil($record/$this->pagesize);         //计算共有几页
            $str.="共有".$contentname." ".$record." ".$utits." 每页显示 ".$this->pagesize." ".
$utits." 第 ".$this->rs->AbsolutePage()." 页/共 ".$pagecount." 页";
            $str.="    ";
            if(!$this->rs->AtFirstPage())
                $str.="<a href=".$_SERVER['PHP_SELF']."?page=".$page."&pages=1".$anothersearchstr." class=".
$class."> 首页</a>";
            else
                $str.="<font color='#555555'>首页</font>";
            $str.=" ";
            if(!$this->rs->AtFirstPage())
                $str.="<a
href=".$_SERVER['PHP_SELF']."?page=".$page."&pages=".($this->rs->AbsolutePage()-1).$anothersearchstr."
class=".$class.">上一页</a>";
            else
                $str.="<font color='#555555'>上一页</font>";
            $str.=" ";
            if(!$this->rs->AtLastPage())
```

```
                    $str.="<a
href=".$_SERVER['PHP_SELF']."?page=".$page."&pages=".($this->rs->AbsolutePage()+1).$anothersearchstr."
class=".$class.">下一页</a>";
            else
                    $str.="<font color='#555555'>下一页</font>";
            $str.=" ";
            if(!$this->rs->AtLastPage())
                    $str.="<a
href=".$_SERVER['PHP_SELF']."?page=".$page."&pages=".$pagecount.$anothersearchstr."  class=".$class.">
尾页</a>";
            else
                    $str.="<font color='#555555'>尾页</font>";
            if(count($this->array)==0 || $this->rs==false)
                    return "";
            else
                   return $str;
      }
}
?>
```

24.6.2　Smarty 模板配置类文件

在 Smarty 模板配置类文件中配置 Smarty 模板文件、临时文件、配置文件等文件路径。system.smarty.inc.php 文件的代码如下：

【例 24.2】 代码位置：光盘\TM\sl\24\system\system.smarty.inc.php

```
<?php
require("smarty/Smarty.class.php");                          //调用 Smarty 类文件
class SmartyProject extends   Smarty{                        //定义类，继承 Smarty 父类
      function SmartyProject(){                              //定义方法，配置 Smarty 模板
            $this->template_dir = "./";                      //指定模板文件存储在根目录下
            $this->compile_dir = "./system/templates_c/";    //指定编译文件存储位置
            $this->config_dir = "./system/configs/";         //指定配置文件存储位置
            $this->cache_dir = "./system/cache/";            //指定缓存文件存储位置
      }
}
?>
```

24.6.3　执行类的实例化文件

在 system.inc.php 文件中，通过 require 语句包含 system.smarty.inc.php 和 system.class.inc.php 文件，执行类的实例化操作，并定义返回对象。完成数据库连接类的实例化后，调用 GetConnId()方法连接数据库。system.inc.php 文件的代码如下：

【例 24.3】 代码位置：光盘\TM\sl\24\system\system.inc.php

```
<?php
require("system.smarty.inc.php");
require("system.class.inc.php");
$connobj=new ConnDB("mysql","localhost","root","root","db_shop",false); //数据库连接类实例化
$conn=$connobj->GetConnId();                          //连接数据库并返回连接标识
$admindb=new AdminDB();                               //数据库操作类实例化
$seppage=new SepPage();                               //分页类实例化
$smarty=new SmartyProject();                          //Smarty 模板配置类实例化

?>
```

24.6.4 表单样式文件

本系统使用了大量的表格和表单，所以预先定义了表单和表格的样式。table.css 样式表文件的代码如下：

【例 24.4】 代码位置：光盘\TM\sl\24\css\table.css

```
<!--  超链接样式  -->
.lk:link {
	text-decoration: none;
	color: #6699CC;
}
.lk:visited {
	text-decoration: none;
	color: #6699CC;
}
.lk:hover {
	text-decoration: none;
}
.lk:active {
	text-decoration: none;
	color: #CCCCFF;
}
<!--  表格首行样式  -->
.first {
	background-color: #6699CC;
	border-bottom-width: 1px;
	border-bottom-style: solid;
	border-bottom-color: #6699CC;
	color: #FFFFFF;
}
<!--  左侧表格样式  -->
.left {
	border-left-width: 1px;
	border-right-width: 1px;
```

```
        border-bottom-width: 1px;
        border-left-style: solid;
        border-right-style: solid;
        border-bottom-style: solid;
        border-left-color: #6699CC;
        border-right-color: #6699CC;
        border-bottom-color: #6699CC;
}
<!--  右侧表格样式  -->
.right {
        border-right-width: 1px;
        border-bottom-width: 1px;
        border-right-style: solid;
        border-bottom-style: solid;
        border-right-color: #6699CC;
        border-bottom-color: #6699CC;
}
<!--  文本框样式  -->
.txt {
        margin: 0px;
        padding: 0px;
        height: 18px;
        width: 100px;
        border: 1px solid #6699CC;
}
<!--  按钮样式  -->
.btn {
        font-size: 12px;
        color: #6699CC;
        background-color: #FFFFFF;
        margin: 0px;
        padding: 0px;
        height: 15px;
        width: 50px;
        border: 1px solid #000000;
}
```

24.7　前台首页设计

视频讲解：光盘\TM\lx\24\前台首页设计.exe

前台首页一般没有多少实质的技术，主要是加载一些功能模块，如登录模块、导航栏模块、公告栏模块等，使浏览者能够了解网站内容和特点。首页的重要之处是要合理地对页面进行布局，既要尽可能地将重点模块显示出来，同时又不能因为页面凌乱无序，而让浏览者无所适从、产生反感。本系统的前台首页 index.php 的运行结果如图 24.16 所示。

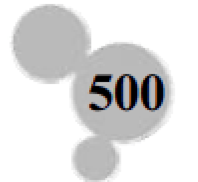
图 24.16　前台首页运行结果

24.7.1　创建 PHP 页

在 index.php 动态页中，应用 include_once()语句包含相应的文件，应用 switch 语句以超链接中参数 page 传递的值为条件进行判断，实现在不同页面之间跳转。index.php 的关键代码如下：

【例 24.5】 代码位置：光盘\TM\sl\24\index.php

```
<?php
session_start();                                          //初始化 SESSION 变量
require_once("system/system.inc.php");                   //获取数据库连接、管理和模板类返回的对象
include_once "public.php";                                //包含公告模块
include_once "links.php";                                 //包含友情链接模块
if($_SESSION["member"]==""){                              //判断 SESSION 变量的值是否为空
     $smarty->assign("member","F");                       //如果为空，则为模板变量赋值为 F
}else{
     $smarty->assign("member",$_SESSION['member']);       //否则将 SESSION 变量的值赋给模板变量
```



```
}
if($_GET["page"]==""){                              //判断变量 page 的值是否为空
      $smarty->assign("page","F");                  //如果为空，为模板变量赋值为 F
}
switch($_GET["page"]){                              //应用 switch 语句，根据条件进行判断，实现页面跳转
        case "hyzx":
           include_once "member.php";               //包含指定的 PHP 文件
           if($_GET['action'] == 'modify'){         //根据 action 的值，指定不同的模板页
                 $smarty->assign("switchs",'modifypwd.html');    //将指定的模板页赋给模板变量
           }else{
                 $smarty->assign("switchs",'membershow.html');
           }
           break;
        case 'allpub':
          include_once 'allpub.php';
                 $smarty->assign("switchs","allpub.html");       //将指定的模板页赋给模板变量
           break;
      …//省略了部分代码
        default:
          include_once 'nominate.php';
                 $smarty->assign("switches","nominate.html");
           include_once 'newhot.php';
                 $smarty->assign("switchs","newhot.html");
      }
      $smarty->display('index.html');                    //指定模板页
?>
```

24.7.2　创建模板页

前台首页的模板文件是 index.html，其中应用 Smarty 的 include 标签调用不同的模板文件，生成静态页面。其关键代码如下：

【例 24.6】 代码位置：光盘\TM\sl\24\index.html

```
<table width="850" border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td colspan="2">{include file="top.html"}</td>
  </tr>
  <tr>
    <td width="216" align="left" valign="top">
     {if $member=="F"}
           {include file="login.html"}
     {else}
           {include file="info.html"}
     {/if}
     {include file="public.html"}
     {include file="links.html"}
    </td>
```

```
      <td width="634" height="720" align="center" valign="top">
       {include file="search.html"}
              {include file=$switchs}
              {if $page==F}
              {include file=$switches}
              {/if}</td>
   </tr>
</table>
<table width="850" border="0" cellspacing="0" cellpadding="0">
       <tr>
       <td>{include file=buttom.html}</td>
      </tr>
</table>
```

下面详细讲解其中几个重点模块的实现过程。读者只要将这几个模块所使用的技术和方法掌握好，其他模块的问题也就迎刃而解了。

说明

本系统的功能较多，结构比较复杂，对于初学者来说学起来可能会比较困难。所以，本书将系统中的各个功能模块所涉及的文件（如 PHP、TPL、CSS、JS 等）尽可能都单独实现。读者在学习其中某个模块时，可以将相关的文件统一放到同一个目录下单独测试。

24.8　登录模块设计

视频讲解：光盘\TM\lx\24\登录模块设计.exe

本模块使用的数据表：tb_user

用户登录模块是会员功能的窗口。匿名用户虽然也可以访问本网站，但只能进行浏览、查询等简单操作，而会员则可以购买商品，并且能享受超低价格。登录模块包括用户注册、用户登录和找回密码 3 部分。

24.8.1　用户注册

用户注册页面的主要功能是新用户注册。如果信息输入完整而且符合要求，则系统会将该用户信息保存到数据库中，否则显示错误原因，以便用户改正。本系统的用户注册模块使用了目前比较流行的 Ajax 技术来验证用户输入的数据，而验证码使用的是 GD2 函数库。本页面的运行结果如图 24.17 所示。

由于篇幅限制，这里只给出有代表性的程序块，即用户名验证和验证码验证两部分，其他信息的验证方法在这两个程序块中都能使用到。

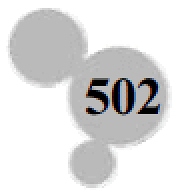


图 24.17　注册模块页面

1. 创建模板

首先来创建模板文件。该模板主要包括注册表单和 div 标签，其中 div 标签是用来提示录入的回馈信息。模板文件 register.html 的核心代码如下：

【例 24.7】 代码位置：光盘\TM\sl\24\register.html

```
<!--  载入 xmlhttp.js 文件和 check.js 文件  -->
<script language="javascript" src="js/createxmlhttp.js"></script>
<script language="javascript" src="js/check.js"></script>
<!--  注册表单部分  -->
<form id="register" name="register" action="reg_chk.php" method="post" onSubmit="return chkinput(this)">
      <!--  用户名验证部分  -->
      <tr>
        <td width="81" height="24"><div align="right">用户名：</div></td>
        <td height="24" colspan="3"> 
            <!--  文本框，将会触发 onBlur 事件，调用 chkname()方法  -->
            <input id = " name " name =" name " type = " text " onBlur = " javascript:chkname( register ) " />
            <!--  隐藏域，默认值为“not”即不允许注册   -->
            <input id = " c_name " name = " c_anme " type = " hidden " value = " not " > <font color = " red "
>*</font></td>
            <!--  div 卷标用于显示对用户名的检测结果   -->
        <td height="24"><div id="name1"><font color="#999999">请输入用户名</font></div></td>
      </tr>
      <tr>
        <td height="24" colspan="2"> 
             <input type="submit" value="提交"/>
            
          <input type="reset" value="重写" /></td>
        <td height="24" colspan="3"><div style="color:#FF0000">带 “*” 号的为必填项</div></td>
      </tr>
```

```
    </form>
</table>
```

这里包含了两个 js 文件：createxmlhttp.js 和 check.js。其中，通过 createxmlhttp.js 文件创建 xmlhttprequest 对象，具体代码可参考光盘中的内容，而 check.js 文件实现了对用户名和验证码的验证过程。下面介绍如何实现用户名验证。

2. 实现用户名验证

用户名验证包括两部分：第一部分是当用户输入名称后，焦点离开当前文本框时所触发的，触发的函数为 chkname(form)。其代码如下：

【例 24.8】 代码位置：光盘\TM\sl\24\js\check.js

```
/*  form 为传入的表单名称，本段代码为 register 表单  */
function chkname(form){
    /*  如果 name 文本域的信息为空  名为 name1 的 div 标签显示如下信息  */
    if(form.name.value==""){
        name1.innerHTML="<font color=#FF0000>请输入用户名！</font>";
    }else{
        /*  否则  获取文本域的值  */
        var user = form.name.value;
        /*  生成 url 链接，将 user 的值传到 chkname.php 页进行判断  */
        var url = "chkname.php?user="+user;
        /*  使用 xmlhttprequest 技术运行页面  */
        xmlhttp.open("GET",url,true);
        xmlhttp.onreadystatechange = function(){
        if(xmlhttp.readyState == 4){
                /*  根据不同的返回值，在 div 标签中输出不同信息  */
                var msg = xmlhttp.responseText;
                if(msg == '3'){
                    name1.innerHTML="<font color=#FF0000>用户名被占用！</font>";
                    return false;
                }else if(msg == '2'){
                    name1.innerHTML="<font color=green>恭喜您，可以注册!</font>";
                /*  如果用户名正确，则将隐藏域的值改为“yes”  */
                    form.c_name.value = "yes";
                }else{
                    name1.innerHTML="<font color=green>未知错误</font>";
                }
            }
        }
        xmlhttp.send(null);
    }
}
```

在该函数中调用了 chkname.php 页，该页在会员登录时也会被调用，所以这里分两种情况，有密码和无密码。无密码为注册验证，当没有返回结果时，说明该用户名可用；而有密码为登录验证，和无密码相反，只有查询记录存在时，才允许登录，并将用户名和用户 ID 存储到 session 中。该页面的



代码如下：

【例 24.9】 代码位置：光盘\TM\sl\24\chkname.php

```
<?php
  session_start()                                              //开启 session 支持
  include_once 'system/system.inc.php';                        //加载数据库连接文件
  $reback = '0';                                               //声明返回变量，初值为‘0’
  $sql = "select * from tb_user where name='".$_GET['user']."'";   //生成 sql 语句
  /*  下面的 SQL 语句加上了对密码的比较，这是用来验证登录的，因为登录验证使用的也是本页面 */
  $password = $_GET['password'];
  if(!empty($password)){
     $sql .= " and password = '".md5($password)."'";           //如果密码不为空，说明是登录验证
  }
  /*  执行 SQL 语句  */
  $rst = $conn->Execute($sql) or die('execute error');         //根据 name 字段查询数据库
  if($rst->RecordCount() == 1){                                //判断 name 是否被使用
     $reback = '3';                                            //被使用返回 3
  /*这里也是对登录页面进行的操作，当用户名和密码输入正确时，将用户名和 ID 存进 session 中  */
  if(!empty($password)){
     $_SESSION['member'] = $rst->fields['name'];
     $_SESSION['id'] = $rst->fields['id'];
     $reback = '1';
     }
  }else{
     $reback = '2';                                            //未被使用返回 2
  }
  echo $reback;
?>
```

上述过程是第一次对用户名进行判断。用户名第二部分是，当用户无视警告信息，单击“提交”按钮后，就会触发<form>表单中的 onsubmit 事件，该事件触发 chkinput()函数，该函数通过隐藏域来判断该注册名称是否合法。其代码如下：

【例 24.10】 代码位置：光盘\TM\sl\24\js\check.js

```
function chkinput(form){
     if(form.name.value==""){                        //判断用户名称文本域是否为空
          form.name.focus();
          return false;
     }
     if(form.c_name.value == "not"){                 //判断隐藏域 c_name 的值是否为 not
          form.name.select();                        //如果是 not，说明该名称未通过验证，返回 false
          return false;
     }
…
}
```

该页面运行结果如图 24.18 所示。



图 24.18　用户名验证过程

3．验证码验证

接下来是验证码验证。先介绍在 register.html 模板中是如何显示验证码的。其代码如下：

【例 24.11】 代码位置：光盘\TM\sl\24\register.html

```
<!--  验证码验证部分  -->
    <tr>
      <td height="24"><div align="right">验证码：</div></td>
      <td width="65" height="24"> 
          <!--  验证码输入框，会触发 onBlur 事件  -->
        <input id="yzm" type="text" name="yzm" size="8" onBlur="javascript:chkyzm(register)" />
          <!--  隐藏域，保存验证码  -->
      <input name="yzm2" type="hidden" value="" /></td>
          <!--  验证码显示区  -->
      <td width="65" align="center" valign="middle"><script>yzm(register)</script></td>
          <!--  在看不清验证码时更换验证图片  -->
      <td width="51"><a href="javascript:code()">看不清</a></td>
          <!--  div 标签，用于显示验证信息  -->
      <td height="24"><div id="yzm1"><font color="#999999">输入验证码</font></div></td>
        </tr>
…
```

通过模板页代码可以看出，模板页是通过调用 yzm()函数来显示验证码的。yzm()函数的代码如下：

【例 24.12】 代码位置：光盘\TM\sl\24\js\check.js

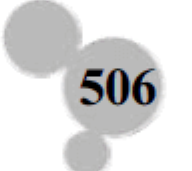
```
function yzm(form){
     var num1=Math.round(Math.random()*10000000);                          //产生随机数
     var num=num1.toString().substr(0,4);                                   //获取前 4 位数
     document.write("<img name=codeimg src='yzm.php?num="+num+"'>");   //输出图片形式的验证码
     form.yzm2.value=num;                                                   //将验证码保存到隐藏域中
}
```

在 yzm()函数内，使用 Math.round()函数生成随机码，并截取前 4 位，然后将验证码提交给 yzm.php 页进行转换，并显示由 yzm.php 生成的 bmp 图片，最后将验证码存储到名为 yzm2d 的隐藏域中，用作验证码检查。yzm.php 页面使用的是 image 函数库。由于不是重点知识，而且使用相对比较复杂，这里不作介绍，感兴趣的读者可参考相关资料。

当用户输入验证码后，可通过 onBlur 事件的 chkyzm()函数来检查。chkyzm()函数的代码如下：

【例 24.13】 代码位置：光盘\TM\sl\24\js\check.js

```
function chkyzm(form){
     if(form.yzm.value==""){
```



```
            <!-- 当输入框信息为空时 -->
            yzm1.innerHTML="<font color=#FF0000>请输入效验码！</font>";
        }else if(form.yzm.value!=form.yzm2.value){
            <!-- 当用户输入与验证码不符时 -->
            yzm1.innerHTML="<font color=#FF0000>效验码输入错误!</font>";
        }else{
            yzm1.innerHTML="<font color=green>输入正确</font>";
        }
}
```

除了验证输入，还可以使用 code()函数重新更换验证码。code()函数的内容几乎和 yzm()函数完全相同，只是显示验证码的方法不一样。code()函数的代码如下：

【例 24.14】 代码位置：光盘\TM\sl\24\js\check.js

```
function code(){
        var num1=Math.round(Math.random()*10000000);
        var num=num1.toString().substr(0,4);
        document.codeimg.src="yzm.php?num="+num;
        form.yzm2.value=num;
}
```

运行结果和用户名验证大同小异，这里不再赘述。

4．保存注册

当用户输入的信息准确无误，系统就会跳转到处理页面 reg_chk.php 来连接数据库，并保存用户数据。保存完毕后，自动使用新用户登录系统。reg_chk.php 页面的代码如下：

【例 24.15】 代码位置：光盘\TM\sl\24\reg_chk.php

```
<?php
        session_start();                                        //开启 session 支持
        include_once 'system/system.inc.php';                   //加载数据库连接文件
        /*   获取表单信息   */
        $name = $_POST['name'];
        $password = md5($_POST['pwd1']);                        //使用 md5()为密码加密
        $question = $_POST['question'];
        $answer = $_POST['answer'];
        $realname = $_POST['realname'];
        $card = $_POST['card'];
        $tel = $_POST['tel'];
        $phone = $_POST['phone'];
        $Email = $_POST['email'];
        $QQ = $_POST['qq'];
        $code = $_POST['code'];
        $address = $_POST['address'];
        $addtime = $conn->DBDate(time());                       //使用 DBDate()函数生成当前时间
        /**************************/
        /*   生成 SQL 语句   */
```

```
	$sql = " insert into tb_user( name, password, question, answer, realname, card, tel, phone, Email, QQ, code, address, addtime ) ";
	$sql .= " values ('$name', '$password', '$question', '$answer', '$realname', '$card', '$tel', '$phone', '$Email', '$QQ', '$code', '$address',$addtime)";
	/*	执行 SQL 语句	*/
	$rst = $conn->execute($sql);
	if($rst == false){
		echo '<script>alert(\'添加失败\');history.back;</script>';
	}else{
		$_SESSION['member'] = $name;
		$_SESSION['id'] = $conn->Insert_ID();
		echo "<script>top.opener.location.reload();alert('注册成功');window.close();</script>";
	}
?>
```

5. 加载模板

因为本系统使用了 Smarty，所以还需要创建一个 PHP 页来加载模板文件。register.php 页面的内容非常简单，页面代码如下：

【例 24.16】 代码位置：光盘\TM\sl\24\register.php

```
<?php
	include_once 'system/system.inc.php';			//载入配置文件
	$smarty->assign('title','新用户注册');			//设置 title
	$smarty->display('register.html');			//显示 register.html
?>
```

6. 创建链接

以上就是用户注册模块的核心代码。当测试没有问题后，最后创建一个“用户注册”超链接。当用户单击前台的用户注册按钮时，系统会调用 js 的 onclick 事件，弹出注册窗口。其代码如下：

【例 24.17】 代码位置：光盘\TM\sl\24\login.html

```
<a href="#" id="login" onclick="reg()">用户注册</a>
```

这里使用到的 js 文件为 js/login.js，调用的函数为 reg()。该函数的代码如下：

【例 24.18】 代码位置：光盘\TM\sl\24\js\login.js

```
function reg(){
window.open("register.php", "_blank", "width=500,height=450",false);//弹出新窗口
}
```

24.8.2 用户登录

除了在注册时直接登录，用户还可以通过其他方式登录。用户登录模块的运行结果如图 24.19 所示。

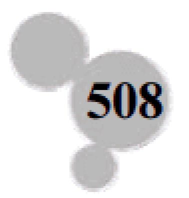




图 24.19　用户登录页面

1. 信息验证

用户登录同样使用的是 Ajax 验证方法。首先来创建页面模板文件 login.html，该文件主要包含一个登录表单。登录表单各元素的属性值及超链接说明如表 24.1 所示。

表 24.1　登录表单各元素属性说明

元 素 名 称	元 素 属 性	元 素 说 明
<form>	id = " login " name = " login " method = " post " action = " # " onsubmit = " return lg(this) "	登录表单可以触发 onsubmit 事件来调用 lg()函数，表单名称为 login
text	id="name" name="name" type="text"	用户名文本框
text	id="password" name="password" type="password"	密码文本框
submit	id="enter" name="enter" type="submit"	登录按钮
<a>看不清	onclick="javascript:code(login)" style=" cursor:hand"	更换验证码图片
hidden	name="check2" type="hidden" value=""	保存验证码的隐藏域
<a>用户注册	id="login" href="#" onclick="reg()"	注册页面超链接
<a>找回密码	id="login" href="#" onclick="found()"	找回密码超链接

通过表 24.1 可以看出，当单击 submit 按钮时，系统将调用 lg()函数。lg()函数包含在 js/login.js 脚本文件内，该函数的代码如下：

【例 24.19】 代码位置：光盘\TM\sl\24\js\login.js

```
function lg(form){
	/*	检验用户名	*/
	if(form.name.value==""){
		alert('请输入用户名');
		form.name.focus();
		return false;
	}
	/*	检验密码	*/
	if(form.password.value == "" || form.password.value.length < 6){
		alert('请输入正确密码');
		form.password.focus();
		return false;
	}
	/*	检验验证码	*/
	if(form.check.value == ""){
		alert('请输入验证码');
		form.check.focus();
		return false;
```

```
    }
    /*    比较验证码    */
    if(form.check.value != form.check2.value){
        form.check.select();
        code(form);
        return false;
    }
    /*    根据输入的用户名和密码生成 url    */
    var user = form.name.value;
    var password = form.password.value;
    var url = "chkname.php?user="+user+"&password="+password;
    /*    使用 xmlhttp 对象来返回验证结果    */
    xmlhttp.open("GET",url,true);
    xmlhttp.onreadystatechange = function(){
    if(xmlhttp.readyState == 4){
            var msg = xmlhttp.responseText;
            /*    如果返回值不等于 2，说明用户名或密码错误    */
            if(msg != '2'){
                alert('用户名或密码错误!!');
                form.password.select();
                form.check.value = '';
                code(form);
            }else{
            /*    如果成功，重新刷新本页    */
                alert('欢迎光临');
                location.reload();
            }
        }
    }
    xmlhttp.send(null);
    return false;
}
```

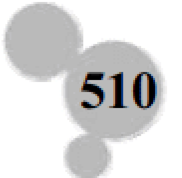
用户名和密码是在 chkname.php 页面中被验证的。chkname.php 在 24.8.1 节中已经介绍过，这里不再赘述。

2. 用户信息

用户登录成功后，在原登录框位置将显示用户信息模块。用户可以通过“会员中心”对自己的信息进行修改，也可以单击“查看购物车”超链接查看购物车商品，当用户离开时可以单击“安全离开”超链接。关于用户信息模块将在 24.9 节中进行讲解。用户信息模块的主要代码如下：

【例 24.20】 代码位置：光盘\TM\sl\24\info.html

```
<!--    显示当前登录用户名    -->
欢迎您：{$member}
<!--    会员中心超链接    -->
<a href="?page=hyzx" id="info" class="lk">会员中心</a>
<!--    查看购物车    -->
<a href="?page=shopcar" class="lk">查看购物车</a>
```



```
<!--   安全离开   -->
<a onclick="javascript:logout()" style="cursor:hand" id="info">安全离开</a>
```

用户信息页面的运行结果如图 24.20 所示。



图 24.20　用户信息页面的运行结果

24.8.3　找回密码

登录模块的最后一个部分就是找回密码。找回密码是根据用户在填写资料时所填写的密保问题和密保答案来实现的。当用户单击“找回密码”超链接时，首先提示用户输入要找回密码的会员名称，然后根据密保问题填写密保答案，最后重新输入密码。找回密码模块的流程如图 24.21 所示。



图 24.21　找回密码模块的流程图

1．创建模板文件

虽然找回密码需要 4 个步骤，但实际上每个步骤使用的都是相同的模板文件和 js 文件，只是被调用的表单和 js 函数略有差别。这里根据不同的文件来分别进行介绍。

该模板文件一共包含了 3 个表单，分别代表了 3 个步骤，其核心代码如下：

【例 24.21】 代码位置：光盘\TM\sl\24\found.html

```
<!--   载入两个 js 脚本文件   -->
<script language="javascript" src="js/createxmlhttp.js"></script>
```

```
<script language="javascript" src="js/found.js"></script>
<!--  第一个 div 标签  -->
<div id="first">
<table width="200" border="0" cellspacing="0" cellpadding="0">
<form id="foundname" name="found" method="post" action="#">
  <tr><td> 找回密码</td></tr>
  <tr><td>会员名称：</td>
     <!--  text 文本域，用于输入要找回密码的会员名称  -->
      <td><input id="user" name="user" type="text" class="txt"></td>
  </tr>
  <tr><td>
<!--  单击“下一步”按钮，触发 onclick 事件来调用 chkname()函数  -->
<input id = " next1 " name = " next1 " type = " button " class = " btn " value = " 下一步 " onClick = " return chkname ( foundname ) "/></td></tr>
</form>
</table>
</div>
<!--  第二个 div 标签，样式为隐藏  -->
<div id="second" style="display:none;">
<table>
<form id="foundanswer" name="found" method="post" action="#">
  <tr><td > 找回密码</td></tr>
  <tr><td>密保问题：</td>
<!--  用于显示密保问题的 div 标签  -->
    <td <div id="question"></div></td></tr>
  <tr><td>密保答案：</td>
<!--  文本域，用于填写密保答案  -->
    <td ><input id="answer" name="answer" type="text" class="txt" /></td></tr>
  <tr>
<!--  单击“下一步”按钮，用来触发 onclick 事件，并调用 chkanswer()函数  -->
    <td><input id = " next2 " name = " next2 " type=" button " class=" btn " value =" 下一步 " onClick = " return chkanswer ( foundanswer ) "></td>
    </tr>
</form>
</table>
</div>
<!--  第 3 个 div 标签，样式也为隐藏，作用是修改密码  -->
<div id='third' style="display:none;">
<table>
<form id="modifypwd" name="found" method="post" action="#">
  <tr><td> 输入密码</td></tr>
  <tr><td>输入密码：</td>
    <td><input id="pwd1" name="pwd1" type="password" class="txt"></td></tr>
  <tr><td>确认密码：</td>
    <td><input id="pwd2" name="pwd2" type="password" class="txt" /></td>
  </tr>
  <tr>
<!--  单击“完成”按钮，调用 ckpwd()函数  -->
    <td><input id = " mod " name = “ mod " type = " button " class = " btn " value = " 完成 " onClick = " return chkpwd (modifypwd) "></td>
```

```
	</tr>
</form>
</table>
</div>
```

可以看出，在上述 3 个表单中，只有一个表单默认情况下是显示的，其他则为隐藏。只有通过调用不同的 js 函数，才可以对其他表单进行操作。

2. 创建 js 脚本文件

found.js 脚本文件包含 3 个函数：chkname()、chkanswer()和 chkpwd()。其中，chkname()函数的作用是检查用户输入的会员名称，如果存在，则使用 xmlhttp 对象去调用生成的 url 进行处理判断。如果该用户存在，则隐藏当前表单，并显示下一个表单，最后输出密保问题。chkname()函数的代码如下：

【例 24.22】 代码位置：光盘\TM\sl\24\js\found.js

```
function chkname(form){
	/*	获取文本框信息	*/
	var user = form.user.value;
	/*	如果为空，则输出提示	*/
	if(user == ''){
		alert('请输入用户名');
		form.user.focus();
		return false;
	}else{
	/*	否则，生成 url，并调用 xmlhttp 对象	*/
		var url = "foundpwd.php?user="+user;
		xmlhttp.open("GET",url,true);
		xmlhttp.onreadystatechange = function(){
		if(xmlhttp.readyState == 4){
				var msg = xmlhttp.responseText;
				/*	如果没有结果，则提示	*/
				if(msg == '0'){
					alert('没有该用户，请重新查找!');
					form.user.select();
					return false;
				}else{
				/*	否则，隐藏当前表单，显示下一个表单，并输出密保问题	*/
					document.getElementById('first').style.display = 'none';
					document.getElementById('second').style.display = '';
					document.getElementById('question').innerHTML = msg;
				}
			}
		}
		xmlhttp.send(null);
	}
}
```

其他两个函数也使用了 xmlhttprequest 对象，实现方法类似，不同之处就是对返回值的处理，chkanswer()函数隐藏了当前表单，显示下一个表单。chkanswer()函数的代码如下：

【例 24.23】 代码位置：光盘\TM\sl\24\js\found.js

```
function chkanswer(form) {
	/*	获取已隐藏表单数据 */
	var user = document.getElementById('user').value;
	/*	获取密保答案	*/
	var answer = form.answer.value;
	/*	表单验证 */
	if(answer == ''){
		…
	}else{
		/*	生成 urlf	*/
		var url = "foundpwd.php?user="+user+"&answer="+answer;
		xmlhttp.open("GET",url,true);
		xmlhttp.onreadystatechange = function(){
			if(xmlhttp.readyState == 4){
				var msg = xmlhttp.responseText;
				if(msg == '0'){
					alert('问题回答错误');
					form.answer.select();
					return false;
				}else{
					document.getElementById('second').style.display = 'none';
					document.getElementById('third').style.display = '';
				}
			}
		}
		xmlhttp.send(null);
	}
}
```

而 chkpwd()函数则提示用户操作状态，如果成功，则关闭当前页。ckpwd()函数的代码如下：

【例 24.24】 代码位置：光盘\TM\sl\24\js\found.js

```
function chkpwd(form){
	/*	获取名称和密码		*/
	var user = document.getElementById('user').value;
	var pwd1 = form.pwd1.value;
	var pwd2 = form.pwd2.value;
	/*	省略部分为表单验证	*/
	var url = "foundpwd.php?user="+user+"&password="+pwd1;
	xmlhttp.open("GET",url,true);
	xmlhttp.onreadystatechange = function(){
		if(xmlhttp.readyState == 4){
			var msg = xmlhttp.responseText;
			if(msg == '1'){
				alert('密码修改成功，请重新登录');
				window.close();
			}
		}
	}
	xmlhttp.send(null);
}
```

3．创建数据处理文件

foundpwd.php 文件的功能是根据用户输入信息来检测数据表中的数据，并根据不同的输入信息返回不同的结果。该文件的代码如下：

【例 24.25】 代码位置：光盘\TM\sl\24\foundpwd.php

```
<?php
	include_once 'system/system.inc.php';				//载入数据库连接文件
	$name= $_GET['user'];							//获取表单值
	$answer = $_GET['answer'];
	$password = $_GET['password'];
	$reback = '0';
	/*	如果变量$answer 和$password 为空，说明用户只输入了会员名称 */
	if(empty($answer) && empty($password)){
		/*	生成查询语句	*/
		$namesql = "select * from tb_user where name = '".$name."'";
		/*	获取查询结果	*/
		$namerst = $conn->execute($namesql);
		/*	如果查询记录为 1，说明输入正确	*/
		if($namerst->recordCount() == 1){
			/*	获取密保问题，并赋给变量$reack	*/
			$question = $namerst->fields['question'];
			$reback = $question;
		}
	/*	如果变量$answer 不为空，说明用户输入了密保答案		*/
	}else if(!empty($answer)){
		/*	根据密保答案和会员名称，生成查询语句	*/
		$answersql = "select * from tb_user where name = '".$name."' and answer = '".$answer."'";
		/*	返回查询结果，并判断是否有返回记录，如果有，说明输入正确		*/
		$answerrst = $conn->execute($answersql);
		if($answerrst->recordCount() == 1){
			$reback = '1';
		}
	/*	如果变量$password 不为空，说明用户输入了新密码 */
	}else if(!empty($password)){
		/*	根据用户名称和新密码更新记录		*/
		$sql = "select * from tb_user where name = '".$name."'";
		$arr = array();
		$rst = $conn->execute($sql);
		if($rst->RecordCount() == 1){
			$arr['password'] = md5($password);
			$updateSQL = $conn->GetUpdateSQL($rst,$arr,true);
			$conn->execute($updateSQL);
			$reback = '1';
		}
	}
	/*	输出返回值		*/
	echo $reback;
?>
```

4．加载模板页

因为所有登录模块的模板都不需要或者只需要传递一两个变量，所以 PHP 加载页的内容比较简单。找回密码页面的代码如下：

【例 24.26】 代码位置：光盘\TM\sl\24\found.php

```
<?php
    include_once 'system/system.inc.php';                    //载入配置文件
    $smarty->assign('title','找回密码');                      //向模板中传递 title 变量
    $smarty->display('found.html');                          //显示 found.html 模板页
?>
```

24.9 会员信息模块设计

视频讲解：光盘\TM\lx\24\会员信息模块设计.exe

本模块使用的数据表：tb_user

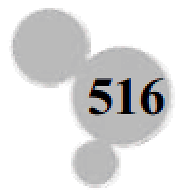
用户登录后，即可看到会员信息模块。在这里，可以查看或修改个人信息及密码、查看购物车和安全退出等。本节只对会员信息模块中的“会员中心”和“安全退出”进行讲解，关于“查看购物车”将在商品模块中进行介绍。

24.9.1 会员中心

当单击“会员中心”超链接时，会回传给当前页一个 page 值，当前页根据这个 page 值来载入 member.php 文件。member.php 页面的运行结果如图 24.22 所示。



图 24.22 会员中心页面的运行结果



1．创建 PHP 页面

与登录模块设计不同，本节首先来创建 PHP 页面。因为该模块中的模板需要使用数据库中的数据及一些动态信息，这些都需要在 PHP 页中先行获取及处理，然后再传给模板页。会员中心页面的代码如下：

【例 24.27】 代码位置：光盘\TM\sl\24\member.php

```
<? php
	session_start();                                         //开启 session 支持
	include_once 'system/system.inc.php';;                   //载入 Smarty 配置文件和数据库连接文件
	/*	查找用户资料	*/
	$sql = 'select * from tb_user where id = '.$_SESSION['id'];
	$rst = $conn->execute($sql);
	$arr = $rst->GetArray();                                 //将结果集输出为数组
	$smarty->assign('arr',$arr[0]);                          //传递数组
?>
```

2．创建模板页

该模块包括查看信息模板及修改密码模板。首先介绍查看信息模板，代码如下：

【例 24.28】 代码位置：光盘\TM\sl\24\membershow.html

```
<!--	显示超链接	-->
<p>{$smarty.session.member}<a href = ' ?page=hyzx ' >查看信息</a><a href = ' ?page = hyzx&action=modify'
id="mem">修改密码</a></p>
<table>
<form id = " member " name = " member " method = " post " action = " modify_info_chk.php " onsubmit = " return
mem( member ) " >
<!--	显示当前会员信息，此处为不可更改信息 -->
  <tr><td colspan="2">{$arr.name}信息（不可更改信息）</td></tr>
<!--	因为格式相同，所以只显示一条，其他省略	-->
  <tr><td>会员编号：</td><td > {$arr.id}</td></tr>
     …
  <tr><td>{$arr.name}信息（可更改信息）</td></tr>
<!--	可以修改的信息，放到表单元素中	-->
  <tr><td>真实姓名：</td>
     <td><input id="realname" name="realname" type="text" value="{$arr.realname}" /></td></tr>
  <tr><td>身份证号：</td>
    <td><input id="card" name="card" type="text" value="{$arr.card}" /></td></tr>
  <tr><td>移动电话：</td>
    <td><input id="tel" name="tel" type="text" value="{$arr.tel}"></td></tr>
  <tr><td>固定电话：</td>
    <td><input id="phone" name="phone" type="text" value="{$arr.phone}" /></td></tr>
  <tr><td>Email：</td>
    <td><input id="email" name="email" type="text" value="{$arr.Email}" /></td></tr>
  <tr><td>QQ 号：</td>
    <td><input id="qq" name="qq" type="text" value="{$arr.QQ}" /></td></tr>
  <tr>
```

```
    <td>邮编：</td>
    <td><input id="code" name="code" type="text" value="{$arr.code}" /></td></tr>
  <tr><td>地址：</td>
    <td><input id="address" name="address" type="text" value="{$arr.address}" /></td></tr>
  <tr>
    <td><input name="enter" type="submit" id="enter" value="修改" />
        <input name="reset" type="reset" id="reset" value="重置" /></td></tr>
</form>
</table>
```

修改密码模板比信息模板要简单一些，因为不需要额外的传值。模板代码如下：

【例 24.29】 代码位置：光盘\TM\sl\24\modifypwd.html

```
<!--  载入脚本文件  -->
<script language="javascript" src="js/member.js"></script>
<p>{$smarty.session.member}<a href='?page=hyzx'>查看信息</a><a href='?page=hyzx&action=modify' >修改密码</a></p>
<table>
<!--  修改密码表单  -->
<form id = " member " name = " member " method = " post " action = " modify_pwd_chk.php " onsubmit = " return pwd (member) ">
  <tr><td >修改密码</td></tr>
  <tr><td >原密码：</td><td><input id="old" name="old" type="password" /></td></tr>
  <tr><td>新密码：</td><td><input id="new1" name="new1" type="password" /></td></tr>
  <tr><td>确认密码：</td><td><input id="new2" name="new2" type="password" /></td></tr>
  <tr><td><input id="enter" name="enter" type="submit" value="修改" /></td></tr>
</form>
</table>
```

3. 创建脚本文件

该模块的脚本文件和用户注册模块类似，都是对信息的合法性进行验证，如信息是否为空、是否符合规范等。这里不再给出代码，有兴趣的读者可以参考 24.8.1 节。

4. 创建处理页

当信息验证通过后，系统将跳转到处理页进行信息处理。本模块处理页分信息修改和密码修改两个页面。首先介绍信息修改页，代码如下：

【例 24.30】 代码位置：光盘\TM\sl\24\modify_info_chk.php

```
<?php
    session_start();                                              //开启 session 支持
    include_once 'system/system.inc.php';                         //载入数据库连接文件
    $sql = 'select * from tb_user where id = '.$_SESSION['id'];  //根据 id 生成 sql 查询语句
    $rst = $conn->execute($sql);                                  //执行 sql 语句，返回查询结果集
    $mod = array();                                               //声明数组
    /*    将修改信息添加到数组中  */
    $mod['realname'] = $_POST['realname'];
    $mod['card'] = $_POST['card'];
```



```
	$mod['tel'] = $_POST['tel'];
	$mod['phone'] = $_POST['phone'];
	$mod['Email'] = $_POST['email'];
	$mod['QQ'] = $_POST['qq'];
	$mod['code'] = $_POST['code'];
	$mod['address'] = $_POST['address'];
	/****************************************/
	$updateSQL = $conn->GetUpdateSQL($rst,$mod);				//生成更新语句
	if($conn->execute($updateSQL))
		echo "<script>alert('修改成功');location=('index.php');</script>";	//修改成功，跳到首页
	else
		echo "<script>alert('修改失败');history.go(-1);</script>";		//否则，返回上一页
?>
```

密码修改页的操作流程也十分类似，只是更新的数组要小得多，只有一个字段。修改密码页的代码如下：

【例 24.31】 代码位置：光盘\TM\sl\24\modify_pwd_chk.php

```
<?php
	session_start();								//开启 session 支持
	include_once 'system/system.inc.php';;				//载入数据库连接文件
	$oldpwd = md5($_POST['old']);						//给密码进行 md5 加密
	/*  根据 id 和 md5 加密后的密码查询用户	*/
	$sql = 'select * from tb_user where id = '.$_SESSION['id'].' and password = \''.$oldpwd.'\'';
	$rst = $conn->execute($sql);						//返回查询结果
	$mod = array();								//创建更新数组
	$mod['password'] = md5($_POST['new1']);				//为新密码进行 md5 加密
	$updateSQL = $conn->GetUpdateSQL($rst,$mod);			//生成更新语句
	if($conn->execute($updateSQL))
		echo "<script>alert('修改成功');location=('index.php');</script>";	//修改成功，返回首页
	else
		echo "<script>alert('修改失败');history.go(-1);</script>";		//否则，返回上一页
?>
```

24.9.2　安全退出

当用户需要离开网站时，可以单击“安全退出”超链接来调用 logout()函数，当用户确认退出后，则跳转到 logout 页面，销毁 session 并回到首页。安全退出所涉及的页面及代码如下：

【例 24.32】 代码位置：光盘\TM\sl\24\js\info.js

```
function logout(){
	if(confirm("确定要退出登录吗？ ")){			//输出选择框，用户可以单击“确认”或“取消”按钮
		window.open('logout.php','_parent','',false);	//如果用户确认退出，则打开 logout.php 页
	}else
	 return false;
}
```

【例 24.33】 代码位置：光盘\TM\sl\24\logout.php

```
<?php
    session_start();                                                          //开启 session 支持
    session_destroy();                                                        //销毁 session
    echo '<script>alert(\'用户已安全退出!\');location=(\'index.php\');</script>';    //回到首页
?>
```

24.10　商品显示模块

视频讲解：光盘\TM\lx\24\商品显示模块.exe

本模块使用的数据表：tb_commo

本系统为用户提供了不同的商品展示方式，包括推荐商品、最新商品、热门商品等，能够使消费者有目的地选购商品。每个展示方式中包括商品的详细信息显示，为用户购买商品提供可靠的依据。本系统商品显示模块的运行结果如图 24.23 所示。



图 24.23　商品展示模块页面

因为推荐商品、最新商品和热门商品的实现方法和过程基本相同，所以本节只讲解推荐商品模块。其他功能相关代码可参见光盘中的源程序。

24.10.1　创建 PHP 页

既然是商品显示模块，那么首先就要取得商品记录。在数据库设计一节中已经了解到，商品字段



isnom 表示商品是否被推荐，如果该字段为 1，即为推荐，否则为不推荐。所以在 PHP 页面中，首先要获取数据库中的 isnom 字段属性为 1 的记录，因为页面限制，所以只返回 4 条记录。首页中的推荐商品页面的代码如下：

【例 24.34】 代码位置：光盘\TM\sl\24\nominate.php

```
<?php
	include_once 'system/system.inc.php';				//连接数据库
	$sql = "select id,name,pics,m_price,v_price from tb_commo where isnom = 1 order by id desc limit 4";
	$nomarr = $admindb->ExecSQL($sql,$conn);		//执行查询语句
	$smarty->assign('nomarr',$nomarr);				//将查询到的数据赋给模板变量
?>
```

24.10.2　创建模板页

记录集数组被传递到模板页后，模板页使用 foreach 标签输出商品信息，并添加相应的操作按钮或链接。模板页中一共有 3 个事件：显示更多商品、查看商品和放入购物车。

- ☑ 当单击“更多商品”超链接时，将会重新加载本页面，并传递一个 page 变量。switch 语句会根据 page 值来显示。
- ☑ 当单击“查看商品”按钮时，将触发 onclick 事件，并将调用 openshowcommo()函数，同时，商品 id 会作为函数的唯一参数被传递进去。
- ☑ 当单击“放入购物车”按钮时，同样会触发 onclick 事件，并调用 buycommo()函数，唯一的参数也是商品的 id。

商品模板页面的代码如下：

【例 24.35】 代码位置：光盘\TM\sl\24\nominate.html

```
<!--  载入 css 样式  -->
<link href="css/nominate.css" rel="stylesheet" type="text/css" />
<link href="css/links.css" rel="stylesheet" type="text/css" />
<!--  载入 js 脚本文件  -->
<script language="javascript" src="js/createxmlhttp.js"></script>
<script language="javascript" src="js/showcommo.js"></script>
<table width="643" border="0" cellpadding="0" cellspacing="0">
	<tr>
		<td colspan="6" background="images/default_14.gif" width="636" height="39" align="right" valign="middle">
<a href="?page=nom" class="lk">&gt;&gt;more&lt;&lt;</a></td>
	</tr>
	<tr>
	{section name=id loop=$nomarr}
		<td width="145" height="185" align="left" valign="top">
	<table width="145" border="0" cellpadding="0" cellspacing="0" >
	<tr>
		<td height="100" align="center" valign="middle"><img src="{$nomarr[id].pics}" width="100" height="80"
alt="{$nomarr[id].name}" style="border: 1px solid #f0f0f0;" ></td>
	</tr>
	<tr>
```

```
            <td height="32" align="center" valign="middle"><input id="showinfo" name="showinfo" type="button" value="" class="showinfo" onclick="openshowcommo({$nomarr[id].id})"/> <input id="buy" name="buy" type="button" value="" class="buy" onclick="return buycommo({$nomarr[id].id})" /></td>
        </tr>
        </table>
        </td>
        {/section}
        </tr>
</table>
```

24.10.3　js 脚本页面

在模板页的 3 个链接中有两个都用到了 js 函数。通过 js 函数，当单击“查看商品”按钮时，系统会弹出一个新的页面，并显示商品的详细信息；当单击“购买”按钮时，该商品将会被放到当前用户的购物车中，如果没有登录用户或商品已添加，则会提示错误信息。js 脚本文件的代码如下：

【例 24.36】 代码位置：光盘\TM\sl\24\js\showcommo.js

```
/*    查看商品信息函数，将打开一个新页面    */
function openshowcommo(key){
        open('showcommo.php?id='+key,'_blank','width=560 height=300',false);
}
/*    将购买商品添加到购物车中，将在下节中讲解    */
function buycommo(key){
        …
}
```

显示更多商品列表和商品详细信息页的实现过程和 24.10.1 节、24.10.2 节类似，只是显示更多字段信息而已，这里不再赘述。24.11 节将重点讲解购物车的实现。

24.11　购物车模块设计

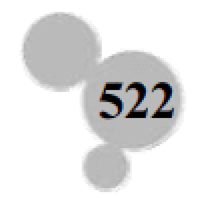
视频讲解：光盘\TM\lx\24\购物车模块设计.exe

本模块使用的数据表：tb_commo、tb_user

购物车在电子商务平台的前台用户端程序是非常关键的一个功能模块。购物车的主要功能是保留用户选择的商品信息，用户可以在购物车内设置选购商品的数量，显示选购商品的总金额，还可以清除选择的全部商品信息，重新选择商品信息。购物车页面运行结果如图 24.24 所示。

我的购物车						
	商品名称	购买数量	市场价格	会员价格	折扣率	合计
□	家庭影院	1	8888 元	6666 元	7.5	6666 元
□	数码相机	1	3200 元	2080 元	6.5	2080 元
全选 反选	删除选择		继续购物	去收银台		共计：8746 元

图 24.24　购物车页面



购物车模块主要实现添加商品、删除商品和更改数量等操作。下面就来一一学习。

24.11.1　添加商品

在商品显示模块中，单击相应商品中的“购买”按钮，即可将物品放到购物车中，并进入到“购物车”页面。在 24.10 节中已经了解到购买操作调用的 buycommo()函数，购买商品的 id 是该函数的唯一参数。下面来看 buycommo()函数的功能结构。函数代码如下：

【例 24.37】 代码位置：光盘\TM\sl\24\js\showcommo.js

```
/*
 *添加商品，同时检查用户是否登录、商品是否重复等
*/
function buycommo(key){
	/*	根据商品 ID，生成 url	*/
	var url = "chklogin.php?key="+key;
	/*	使用 xmlhttp 对象调用 chklogin.php 页	*/
	xmlhttp.open("GET",url,true);
	xmlhttp.onreadystatechange = function(){
		if(xmlhttp.readyState == 4){
				var msg = xmlhttp.responseText;
				/*	用户没有登录	*/
				if(msg == '2'){
					alert('请您先登录');
					return false;
				}else if(msg == '3'){
				/*	商品已添加	*/
					alert('该商品已添加');
					return false;
				}else{
				/*	显示购物车	*/
					location='index.php?page=shopcar';
				}
			}
	}
	xmlhttp.send(null);
```

可以看到，buycommo()函数通过 xmlhttp 对象调用了 chklogin.php 页，并根据回传值作出相应的处理。chklogin.php 页才是将商品添加到购物车中的操作页。chklogin.php 页的代码如下：

【例 24.38】 代码位置：光盘\TM\sl\24\chklogin.php

```
<?php
	session_start();								//开启 session 支持
	/**
	 *	1 表示添加成功
	 *	2 表示用户没有登录
	 *	3 表示商品已添加过
```

```
 *   4 表示添加时出现错误
 *   5 表示没有商品添加
 */
include_once 'system/system.inc.php';                               //载入数据库连接文件
$reback = '0';                                                      //声明返回值
if(empty($_SESSION['member'])){                                     //如果 session 值为空，说明没有登录
	$reback = '2';                                                  //参见头注释
}else{
	$key = $_GET['key'];                                            //获取商品 id
	if($key == ''){
		$reback = '5';
	}else{
		$id = (int)$_SESSION['id'];                                 //获取登录会员 id 号
		$boo = false;                                               //设置一个布尔变量，初始为 false
		$addshop = array();                                         //数据库更新数组
		$sql = "select id,shopping from tb_user where id = ".$id;   //生成当前用户的查询语句
		$rst = $conn->execute($sql);                                //返回当前用户记录集
		$shopcont = $rst->fields['shopping'];                       //找到 shopping 字段
		if(!empty($shopcont)){                                      /如果字段不为空，说明已有商品
			$arr = explode('@',$shopcont);                          //将字段用“@”拆分并另存数组
			foreach($arr as $value){                                //循环输出数组内容
				if($key == $value[0]){                              //判断需要添加的商品是否存在
					$reback = '3';                                  //参见头注释
					$boo = true;                                    //同时布尔变量变为 true
					break;                                          //停止循环
				}
			}
			if(!$boo){                                              //如果布尔值没有改变，再添加商品
				/*   将商品数量默认设为 1，并连同商品 id 保存到字串中*/
				$shopcont .= '@'.$key.',1';
				/*   将商品字串保存到更新数组中   */
				$addshop['shopping'] = $shopcont;
				/*   生成更新语句   */
				$updateSQL = $conn->GetUpdateSQL($rst,$addshop);
				/*   更新数据表     */
				if(false == $conn->execute($updateSQL)){
					$reback = '4';
				}else{
					$reback = '1';
				}
			}
		}else{
			/*   如果 shopping 字段为空，则直接更新    */
			$tmparr = $key.",1";
			$addshop['shopping'] = $tmparr;
			$updateSQL = $conn->GetUpdateSQL($rst,$addshop);
			if(false == $conn->execute($updateSQL)){
				$reback = '4';
			}else{
```

```
                        $reback = '1';
                    }
                }
            }
        }
        echo $reback;
?>
```

通过分析上述代码可知，shopping 字段保存的是购物车中的商品信息，一条商品信息包括两部分，即商品 id 和商品数量，其中商品数量默认为 1。两部分之间使用逗号“,”分隔，如果添加多个商品，则每个商品之间使用“@”分隔。

24.11.2　显示购物车

商品添加后，系统直接跳转到购物车页面，显示添加的商品信息。购物车页面同样分 PHP 代码页和 Smarty 模板页。在 PHP 代码页中，先读取 tb_user 数据表中 shopping 字段的内容，如果字段为空，则输出“无商品信息”文本；如果数据库中有数据，则使用循环，将商品信息保存到数组中，再传给模板页。购物车页面的代码如下：

【例 24.39】 代码位置：光盘\TM\sl\24\myshopcar.php

```
<?php
    /*    载入数据库连接文件和 Smarty 配置文件  */
    include_once 'system/system.inc.php';
    /*    根据登录会员的用户名生成查询语句 */
    $sql1 = "select id,shopping from tb_user where name ='".$_SESSION['member']."'";
    /*    执行查询语句返回结果集  */
    $rst = $conn->execute($sql1);
    /*    如果字段为空，直接输出“无商品信息”，并结束运行    */
    if($rst->fields['shopping'] == ''){
        echo "<p>";
        echo '购物车中暂时没有商品!';
        exit();
    }
    /*    如果有结果集，则将结果集另存为数组    */
    $tmparr = $rst->GetAssoc();
    $commarr = array();
    /*    输出 shopping 字段值            */
    foreach($tmparr as  $value){
        /*    使用“@”进行分隔，得到数组      */
        $tmpnum = explode('@',$value);
        /*    得到购买商品种类    */
        $shopnum = count($tmpnum);
        /*    商品总价值      */
        $sum = 0;
        /*    数组循环  */
```

```
            foreach($tmpnum as $key => $vl){
                /*    使用逗号“,”拆分单一的商品信息        */
                $s_commo = explode(',',$vl);
                /*    根据保存的商品信息查找商品    */
                $sql2 = "select id,name,m_price,fold,v_price from tb_commo";
                $commsql = $sql2." where id = ".$s_commo[0];
                /*    返回结果集      */
                $commrst = $conn->execute($commsql);
                /*    将结果集另存数组    */
                $arr = $commrst->GetArray();
                /*    将需要显示的信息处理后保存到数组中      */
                $arr[0]['num'] = $s_commo[1];
                $arr[0]['total'] = $s_commo[1]*$arr[0]['v_price'];
                $sum += $arr[0]['total'];
                $commarr[$key] = $arr[0];
            }
        }
        /*    将得到的两个数组和总消费金额传递给模板      */
        $smarty->assign('shoparr',$shopnum);
        $smarty->assign('commarr',$commarr);
        $smarty->assign('sum',$sum);
        $smarty->assign('title','我的购物车');
?>
```

商品的模板页不仅要负责将传入的商品信息数组显示出来，而且还要提供可以对商品进行修改、删除等操作的事件接口。模板页的代码如下：

【例 24.40】 代码位置：光盘\TM\sl\24\myshopcar.html

```
<!--   载入使用到的 js 脚本文件   -->
<script language="javascript" src="js/createxmlhttp.js"></script>
<script language="javascript" src="js/shopcar.js"></script>
<table>
<!--   购物车表单   -->
<form id="myshopcar" name="myshopcar" method="post" action="#">
  <tr><td>我的购物车</td></tr>
  <tr><td > </td><td>商品名称</td><td>购买数量</td><td>市场价格</td>
     <td>会员价格</td><td>折扣率</td><td>合计</td></tr>
<!--   使用 foreach 表单循环输出商品种类   -->
{foreach key=key item=item from=$commarr}
<!--   删除复选框，名称用数组表示，每个复选框的值为商品的 id 号   -->
  <tr><td><input id="chk" name="chk[]" type="checkbox" value="{$item.id}"></td>
<!--   显示该商品的名称   -->
            <td><div id = "c_name{$key}">  {$item.name}</div></td>
     <td>
<!--   商品数量文本框   每个复选框的名称为“cnum”+key 值   -->
            <input id = " cnum{$key} " name = " cnum{$key} " type = " text " value = " {$item.num} " onkeyup = "
cvp ( {$key},{$item.v_price},{$shoparr})"></td>
<!--   商品的其他信息   -->
     <td><div id="m_price{$key}"> {$item.m_price}</div></td>
```

```
        <td><div id="v_price{$key}"> {$item.v_price}</div></td>
        <td><div id="fold{$key}"> {$item.fold}</div></td>
        <td><div id="total{$key}"> {$item.total}</div></td>
    </tr>
<!--  循环结束  -->
{/foreach}
    <tr>
        <td colspan="3">
<!--  全选、反选和删除操作  -->
            <a href="#" onclick="return alldel(myshopcar)">全选</a>
            <a href="#" onclick="return overdel(myshopcar);">反选</a>
            <input type="button" value="删除选择" onClick = 'return del(myshopcar);'></td>
        <td>
<!--  继续购物  -->
                <input type="button" value="继续购物" onclick="return conshop(myshopcar)" /></td>
        <td>
<!--  将当前用户名保存到隐藏域中  -->
                <input id="uid" name="uid" type="hidden" value="{$smarty.session.member}" >
<!--  结算，去收银台  -->
                <input type="button" class="btn" value="去收银台" onclick="return formset(form)" /></td>
        <td colspan="2"><div id='sum'>共计：{$sum} 元</div></td>
        </tr>
</form>
</table>
```

24.11.3　更改商品数量

对于新添加的商品，默认的购买数量为 1，在购物车页面可以对商品的数量进行修改。当商品数量发生变化时商品的合计金额和商品总金额会自动发生改变，该功能是通过触发 text 文本域的 onkeyup 事件来调用 cvp()函数实现的。cvp()函数有 3 个参数，分别是商品 id、商品单价和商品类别。

首先通过商品的 id 可以得到要修改商品的相关表单和标签属性，然后通过商品单价和输入的商品数量计算该商品的合计金额，接着使用 for 循环得到其他商品的合计金额，最后将所有的合计金额累加，并输出到购物车页面。cvp()函数的代码如下：

【例 24.41】 代码位置：光盘\TM\sl\24\js\shopcar.js

```
/* 自动刷新总金额
 * key：商品 id
 * vpr：商品单价
 * shoparr：商品种类数
*/
function cvp(key,vpr,shoparr){
        var n_pre = 'total';                                    //div 标签的 id 前缀
        var num = 'cnum'+key.toString();                        //根据 key 值生成文本域的 id 值
        var total = n_pre+key.toString();                       //根据 key 值生成 div 标签的 id 值
        var t_number = document.getElementById(num).value;      //获取输入的商品数量
        var ttl = t_number * vpr;                               //根据商品数量和单价，计算商品金额
```

```
        document.getElementById(total).innerHTML = ttl;                    //更新商品总金额
        var sm = 0;                                                        //所有商品的总金额，初始为 0
        for(var i = 0; i < shoparr; i++){                                  //根据商品种类数，循环得到其他商品金额
                var aaa = document.getElementById(n_pre+i.toString()).innerText;
                sm += parseInt(aaa);                                       //将所有商品金额累加
        }
        document.getElementById('sum').innerHTML = '共计：'+sm+' 元';
}
```

这里所更改的商品数量，并没有被保存到数据库中，如果希望保存，可单击“继续购物”按钮，即将商品数量更新到数据库中。该功能将在 24.11.5 节中实现。

24.11.4　删除商品

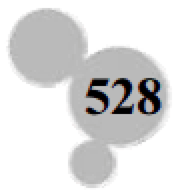
当对添加的商品不满意时，可以对商品进行删除操作。操作流程为：首先选中要删除的商品前面的复选框，如果全部删除，则可以单击“全选”按钮或“反选”按钮，然后单击“删除选择”按钮，在弹出的警告框中单击“确定”按钮，商品将被全部删除。删除商品的页面结果如图 24.25 所示。



图 24.25　删除商品流程

所有的删除操作都是通过 js 脚本文件 shopcar.js 来实现的，相关的函数包括 alldel()函数、overdel()函数和 del()函数。下面就分别介绍这 3 个函数的实现过程。

alldel()函数和 overdel()函数实现的原理比较简单，通过触发 onclick 事件来改变复选框的选中状态。函数实例代码如下：



【例 24.42】 代码位置：光盘\TM\sl\24\js\shopcar.js

```
/*    全部选择  */
function alldel(form){
        var leng = form.chk.length;                        //获取复选框数量
        if(leng==undefined){                               //如果等于 undefined，说明只有一个复选框
            if(!form.chk.checked)
                    form.chk.checked=true;                 //将复选框置于选中状态
         }else{                                            //undefined，说明有多个复选框
           for( var i = 0; i < leng; i++)                  //使用 for 循环，将所有复选框选中
             {
                    if(!form.chk[i].checked)
                            form.chk[i].checked = true;
             }
         }
        return false;
}
/*    反选，就是将选中的复选框取消选中，而未被选中的复选框则被选中 */
function overdel(form){
         var leng = form.chk.length;                       //获取复选框个数
         if(leng==undefined){                              //如果 leng 为 undefined，说明只有一个复选框
            if(!form.chk.checked)
                    form.chk.checked=true;
              else
                    form.chk.checked=false;
         }else{                                            //否则，说明有多个复选框
           for( var i = 0; i < leng; i++)                  //使用 for 循环，对所有复选框操作
             {
                    if(!form.chk[i].checked)               //根据复选框 checked 属性的情况，进行反向选择
                            form.chk[i].checked = true;
                    else
                            form.chk[i].checked = false;
             }
         }
        return false;
}
```

使用 alldel()或 overdel()选中复选框后，即可调用 del()函数来实现删除功能。del()函数首先使用 for 循环，将被选中的复选框的 value 值取出并存成数组，然后根据数组生成 url，并使用 xmlhttp 对象调用这个 url，当处理完毕后，根据返回值弹出提示或刷新本页。该函数的代码如下：

【例 24.43】 代码位置：光盘\TM\sl\24\js\shopcar.js

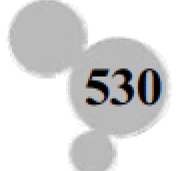
```
/*  删除记录   */
function del(form){
        if(!window.confirm('是否要删除数据??')){

        }else{
                var leng = form.chk.length;                //返回复选框的长度
                if(leng==undefined){
                        if(!form.chk.checked){
```

```
                        alert('请选取要删除数据!');
                }else{                                          //如果只有一个复选框，且处于选中状态
                        rd = form.chk.value;                    //将复选框的 value 值直接赋给变量 rd
                        var url = 'delshop.php?rd='+rd;         //根据 rd 生成 url
                        xmlhttp.open("GET",url,true);           //调用 xmlhttp 对象
                        xmlhttp.onreadystatechange = delnow;    //调用 delnow()函数
                        xmlhttp.send(null);
                }
        }else{                                                  //如果复选框为多个
                var rd=new Array();                             //声明数组
                var j = 0;
                for( var i = 0; i < leng; i++)                  //循环检查复选框状态
                {
                        if(form.chk[i].checked){
                                rd[j++] = form.chk[i].value;    //将被选中的复选框的 value 值存到 rd 内
                        }
                }
                if(rd == ''){
                        alert('请选取要删除数据!');
                }else{
                        var url = "delshop.php?rd="+rd;         //解释同上
                        xmlhttp.open("GET",url,true);
                        xmlhttp.onreadystatechange = delnow;
                        xmlhttp.send(null);
                }
        }
    }
    return false;
}
/*    显示状态  */
function delnow(){
    if(xmlhttp.readyState == 4){
        if(xmlhttp.status == 200){
                var msg = xmlhttp.responseText;                 //获取 xmlhttp 对象返回的文本值
                if(msg != '1'){
                        alert('删除失败');                      //如果为 1，说明删除失败
                }else{
                        alert('删除成功');
                        location=('?page=shopcar');             //否则，说明删除成功，刷新购物车
                }
        }
    }
}
```

24.11.5 保存购物车

当用户希望保存商品数量时，可以单击“继续购物”按钮。这时，系统将触发 onclick 事件来调用 conshop()函数来保存数据，该函数有一个参数，就是当前表单的名称。在 conshop()函数内，根据复选



框和商品数量文本域，生成两个数组 fst 和 snd，分别保存商品 id 和商品数量。这里要注意，两个数组的值是要相互对应的，如商品 1 的 id 保存到 fst[1]中，那么商品 1 的数量就要保存到 snd[1]中，然后根据这两个数组生成一个 url，使用 xmlhttprequest 对象调用 url，最后根据回传信息作出相应的判断。conshop()函数的代码如下：

【例 24.44】 代码位置：光盘\TM\sl\24\js\shopcar.js

```
/*	更改商品数量	*/
function conshop(form){
	var n_pre = 'cnum';						//商品数量文本域前缀
	var lang = form.chk.length;					//获取复选框的数量
/*	如果只有一个复选框，那么将商品 id 和商品数量直接保存到变量中	*/
	if(lang == undefined){
		var fst = form.chk.value;
		var snd = form.cnum0.value;
/*	否则，将商品 id 和对应的商品数量保存到两个相应的数组中	*/
	}else{
		var fst= new Array();					//商品 id 数组
		var snd = new Array();					//商品数量数组
/*	循环获取复选框的 value 值和商品数量文本框的 value 值 */
		for(var i = 0; i < lang; i++){
			var nm = n_pre+i.toString();
			var stmp = document.getElementById(nm).value;
/*	对商品数量文本框判断，不允许为空，而且不允许为非数字输入	*/
			if(stmp	== '' || isNaN(stmp)){
				alert('不允许为空，必须为数字');
				document.getElementById(nm).select();
				return false;
			}
			snd[i] = stmp;
			var ftmp = form.chk[i].value;
			fst[i] = ftmp;
		}
	}
	var url = 'changecar.php?fst='+fst+'&snd='+snd;		//生成 url
	xmlhttp.open("GET",url,true);					//调用 xmlhttp 对象
	xmlhttp.onreadystatechange = updatecar;			//调用 updatecar()函数
	xmlhttp.send(null);
}
/*	对 xmlhttp 对象的返回值进行处理	*/
function updatecar(){
	if(xmlhttp.readyState == 4){
		var msg = xmlhttp.responseText;
			if(msg == '1'){
				location='index.php';				//如果操作成功，返回首页
			}else{
				alert('操作失败');				//否则，提示错误
			}
	}
}
```

在函数中调用的 changecar.php 页为数据处理页，该页将商品 id 和商品数量进行重新排列，并保存到 shopping 字段内。该页面的代码如下：

【例 24.45】 代码位置：光盘\TM\sl\24\changecar.php

```
<?php
	include_once 'system/system.inc.php';								//连接数据库
	$sql = 'select id,shopping from tb_user where id = '.(int)$_SESSION['id'];	//生成 sql 查询语句
	$fst = $_GET['fst'];												//获取商品 id
	$snd = $_GET['snd'];												//获取商品数量
	$reback = '0';													//设置返回值，默认为 0
	$rst = $conn->execute($sql);										//执行查询语句，返回查询结果集
	$changecar = array();												//创建更新数组
	if($fst != '' and $snd != ''){
		/*	使用逗号“,”将商品 id 和商品数量分隔，并分别保存到两个临时数组中	*/
		$farr = explode(',',$fst);
		$sarr = explode(',',$snd);
		/*	临时数组	*/
		$upcar = array();
		/*	使用 for 循环，将商品 id 和商品数量一一对应，重新组合 */
		for($i = 0; $i < count($farr); $i++){
			$upcar[$i] = $farr[$i].','.$sarr[$i];
		}
		/*	如果数组大于 1，说明有多个商品，使用“@”将不同商品分隔开		*/
		if(count($farr) > 1){
			$changecar['shopping'] = implode('@',$upcar);
		}else{
		/*	如果只有一个商品，则直接保存到更新数组中	*/
			 $changecar['shopping'] = $upcar[0];
		}
		/*	生成更新语句	*/
			$UpdateSql = $conn->GetUpdateSQL($rst,$changecar,true);
		/*	更新数据库	*/
			if(false == $conn->execute($UpdateSql)){
				$reback = '2';
			}else{
				$reback = '1';
			}
	}
	echo $reback;
?>
```

24.12 收银台模块设计

视频讲解：光盘\TM\lx\24\收银台模块设计.exe

本模块使用的数据表：tb_commo、tb_user、tb_form

当用户停止浏览商品准备结账时，可以通过单击购物车页面中的“去收银台”按钮来实现，该按



钮将触发 onclick 事件调用 formset()函数来显示订单，当用户提交订单后，系统将订单保存到数据表 tb_form 中，同时清空购物车，并显示订单信息提醒用户记录订单号。当货款发出后，还可以对订单进行查询。收银台页面的运行结果如图 24.26 所示。



图 24.26　收银台页面运行结果

本节所涉及的页面有显示订单（formset()函数）、填写订单（settle.php、settle.html）、处理订单（settle_chk.php）、反馈订单（forminfo.php、forminfo.html）和查询订单 5 部分。首先来介绍 formset()函数。

24.12.1　显示订单

formset()函数的作用是将商品信息整理后，通过 open 方法打开 settle.php 页来显示订单，并将整理后的商品信息一并传过去。formset()函数的代码如下：

【例 24.46】 代码位置：光盘\TM\sl\24\js\shopcar.js

```
function formset(form){
var uid = form.uid.value;                                          //获取订单提交人的名称
var n_pre = 'cnum';                                                //商品数量文本前缀
	var lang = form.chk.length;                                    //复选框个数
	/*	如果复选框只有一个 */
	if(lang == undefined){
		var fst = form.chk.value;                                  //商品 id
		var snd = form.cnum0.value;                                //商品数量
	/*	当有多个复选框时	*/
	}else{
		var fst= new Array();
		var snd = new Array();
	/*	使用 for 循环，获取商品数量和商品 id	*/
		for(var i = 0; i < lang; i++){
			var nm = n_pre+i.toString();
			var stmp = document.getElementById(nm).value;
			if(stmp	== '' || isNaN(stmp)){
				alert('不允许为空，必须为数字');
				document.getElementById(nm).select();
				return false;
			}
```

```
                        snd[i] = stmp;
                        var ftmp = form.chk[i].value;
                        fst[i] = ftmp;
                }

        }
        /*    使用 open 方法，打开 settle.php 页，并将当前用户会员名称、商品 id 和商品数量的值传过去 */
        open('settle.php?uid='+uid+'&fst='+fst+'&snd='+snd,'_blank','width=420 height=220',false);
}
```

说明

因为 open 方法使用了_blank 参数来打开一个新的页面，session 值传不过去，所以这里使用隐藏域来传递用户名称。

24.12.2 填写订单

settle.php 直接将接收过来的值传给 settle.html 模板，并载入 settle.html 模板。settle.php 页面的代码如下：

【例 24.47】 代码位置：光盘\TM\sl\24\settle.php

```
<?php
        include_once 'system/system.inc.php';           //载入 Smarty 配置文件
        $fst = $_GET['fst'];                            //获取 GET 属性值
        $snd = $_GET['snd'];
        $uid = $_GET['uid'];
        $smarty->assign('title','结账');                 //将标题、数组等数据传递给模板
        $smarty->assign('fst',$fst);
        $smarty->assign('snd',$snd);
        $smarty->assign('uid',$uid);
        $smarty->display('settle.html');                //显示 settle.html 模板文件
?>
```

settle.html 模板显示一个表单，这个表单的内容需要用户来填写，包括收货人、联系电话等信息。而从 PHP 页传过来的几个变量则被保存到隐藏域以传递到处理页。settle.html 模板的表单元素及属性值如表 24.2 所示。

表 24.2 settle.html 模板的表单元素及属性值

元 素 名 称	元 素 属 性	元 素 说 明
<form>	id="buyform"name="buyform"method="post"action="settle_chk.php"	表单名为 buyform，提交页为 settle_chk.php
text	id="taker" name="taker" type="text"	收货人文本域
text	id="code" name="code" type="text"	邮编文本域
text	id="tel" name="tel" type="text"	电话文本域
Text	id="address" name="address" type="text"	地址文本域



续表

元 素 名 称	元 素 属 性	元 素 说 明
<select>	<select id="del" name="del"> <option value="平邮">平邮</option> <option value="快递">快递</option> <option value="送货上门">送货上门</option> </select>	送货方式
<select>	<select id="pay" name="pay"> <option value="银行转账">银行转账</option> <option value="邮局汇款">邮局汇款</option> <option value="支付宝">支付宝</option> </select>	付款方式
<hidden>	id="fst" name="fst" type="hidden" value="{$fst}"	保存商品 id
<hidden>	id="snd" name="snd" type="hidden" value="{$snd}"	保存商品数量
<hidden>	id="uid" name="uid" type="hidden" value="{$uid}"	保存当前用户
<submit>	id="enter" name="enter" type="submit" value="提交订单"	"提交订单"按钮

24.12.3　处理订单

用户添加表单信息后，单击"提交"按钮，将会调用处理页 settle_chk.php 来处理表单数据。处理页接收到数据后，根据用户提交的商品信息，重新查找数据表 tb_commo，并从数据表中提取商品的一些信息保存到数组中，然后处理页将数组作为一条记录添加到表 tb_form 内。数据添加成功的同时，处理页会根据 uid 找到该用户，将 shopping 字段清空，最后调用 forminfo.php 页来显示新添加的订单信息。settle_chk.php 页的代码如下：

【例 24.48】 代码位置：光盘\TM\sl\24\settle_chk.php

```
<?php
/*     获取表单值，并存储到添加数组中     */
       include_once 'system/system.inc.php';
       $sql = "select * from tb_form where id = -1";
       $rst = $conn->execute($sql);
       $addform = array();
       $addform['vendee'] = $_POST['uid'];                                    //提交订单用户
       $addform['commo_id'] = $_POST['fst'];                                  //购买商品 id
       $addform['commo_num'] = $_POST['snd'];                                 //购买商品数量
/*     根据商品 id 和商品数量，获取所需数据     */
       $addform['formid'] = time()                                            //订单号
       $tmpid = explode(',',$addform['commo_id']);                            //将商品 id 存为数组
       $tmpnm = explode(',',$addform['commo_num']);                           //将商品数量存为数组
       $number = count($tmpid);                                               //获取商品数组长度
       /*     当购买商品种类大于 1 时  */
       if($number >1){
              $tmpna = array();                                               //商品名称数组
```

```
        $tmpvp = array();                                                    //商品单价数组
        $tmpfd = array();                                                    //商品折扣数组
        $tmptt = 0;                                                          //商品总额数组
        /*    使用 for 循环获取各种商品的信息    */
        for($i = 0; $i < $number; $i++){
            /*    生成查询语句，执行 sql 语句，最后返回查询结果集 */
            $tmpsql = "select name,v_price,fold from tb_commo where id = ".$tmpid[$i];
            $tmprst = $conn->execute($tmpsql);
            /*    将每种商品的信息放到对应的数组中 */
            $tmpna[$i] = $tmprst->fields['name'];
            $tmpvp[$i] = $tmprst->fields['v_price'];
            $tmpfd[$i] = $tmprst->fields['fold'];
            $tmptt += $tmprst->fields['v_price'] * $tmpnm[$i];
        }
        /*    将获取的信息数组分别生成长字符串，数组值之间使用逗号“,”分隔    */
        $addform['commo_name'] = implode(',',$tmpna);                        //商品名称
        $addform['agoprice'] = implode(',',$tmpvp);                          //商品价格
        $addform['fold'] = implode(',',$tmpfd);                              //商品折扣
        $addform['total'] = $tmptt;                                          //商品总额
    /*    当购买一种商品时    */
    }else if($number == 1){
        $tmpsql = "select name,v_price,fold from tb_commo where id = ".$tmpid[0];
        $tmprst = $conn->execute($tmpsql);

        $addform['commo_name'] = $tmprst->fields['name'];
        $addform['agoprice'] = $tmprst->fields['v_price'];
        $addform['fold'] = $tmprst->fields['fold'];
        $addform['total'] = $tmprst->fields['v_price'] * $tmpnm[0];
    /*    没有商品时    */
    }else{
        echo 'error';
        exit();
    }
    /*    将表单的其他信息存入添加数组    */
    $addform['taker'] = $_POST['taker'];                                     //收货人
    $addform['code'] = $_POST['code'];                                       //邮编
    $addform['tel'] = $_POST['tel'];                                         //联系电话
    $addform['address'] = $_POST['address'];                                 //地址
    $addform['del_method'] = $_POST['del'];                                  //送货方式
    $addform['pay_method'] = $_POST['pay'];                                  //付款方式
    $addform['state']  = '0';                                                //订单状态，0 为未处理
    $InsertSQL = $conn->GetInsertSQL($rst,$addform);                         //生成 sql 添加语句
    if(false == $conn->execute($InsertSQL)){
        /*    如果购买失败，返回上层  */
        echo "<script>alert('购买失败');history.back;</script>";
    }else{
        /*    如果成功，删除购物车信息，并显示订单信息  */
        $updsql = "select * from tb_user where name = '".$_POST['uid']."'";
        $updrst = $conn->execute($updsql);
        $arr = array();
```

```
            $arr['consume'] = $addform['total'];
            $arr['shopping'] = '';
            $UpdateSQL = $conn->GetUpdateSQL($updrst,$arr);
            $conn->execute($UpdateSQL);
            echo "<script>top.opener.location.reload();</script>";
            echo "<script>open('forminfo.php?fid=$fid','_blank','width=600 height=450',false);</script>";
            echo “<script>window.close(); </script>";
      }
?>
```

24.12.4　反馈订单

订单提交成功后，forminfo.php 页会将新添加的订单反馈给用户，并提醒用户保存订单号，以便根据订单号来付款、查询及处理问题。反馈订单页面的运行结果如图 24.27 所示。

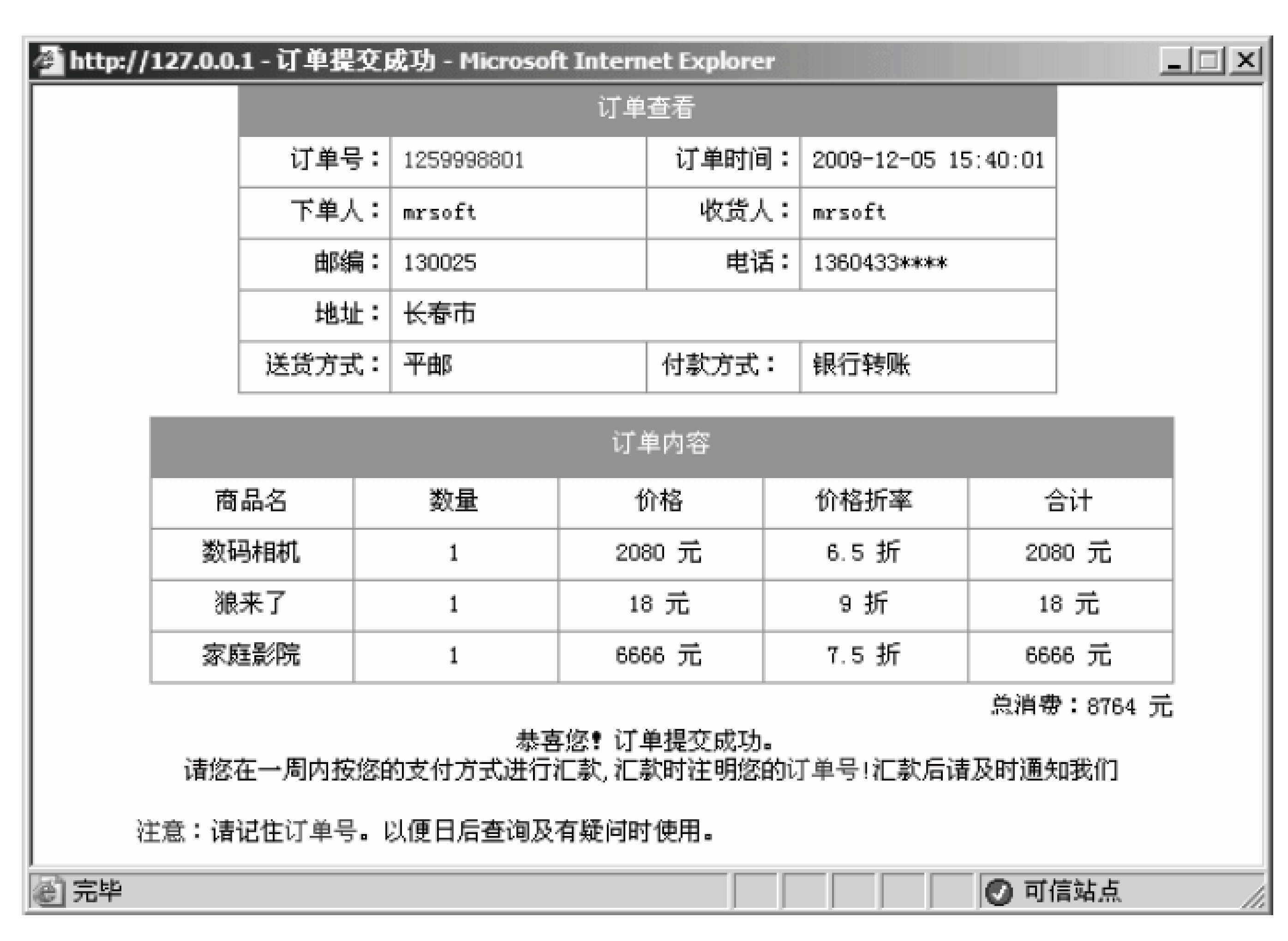


图 24.27　反馈订单页面的运行结果

forminfo.php 页的代码如下：

【例 24.49】　代码位置：光盘\TM\sl\24\forminfo.php

```
<?php
    include_once 'system/system.inc.php';                    //载入数据库连接文件和 Smarty 模板文件
    $id = $_GET['fid'];                                      //获取订单 id
    $sql = "select * from tb_form where id = ".$id;          //根据 id 生成查询语句
    $rst = $conn->execute($sql);                             //执行 sql 语句，返回结果集
    $formarr=$rst->GetArray();                               //将结果集存储为数组
    $commname = explode(',',$formarr[0]['commo_name']);      //将商品名称存为数组
    $commnumber = explode(',',$formarr[0]['commo_num']);     //将商品数量存为数组
    $commagoprice = explode(',',$formarr[0]['agoprice']);    //将商品单价存为数组
    $commfold = explode(',',$formarr[0]['fold']);            //将商品折扣率存为数组
    /*    将各个数组传给模板，最后显示模板        */
```

```
	$smarty->assign('formarr',$formarr[0]);
	$smarty->assign('commname',$commname);
	$smarty->assign('commnumber',$commnumber);
	$smarty->assign('commagoprice',$commagoprice);
	$smarty->assign('commfold',$commfold);
	$smarty->assign('title','订单提交成功');
	$smarty->display('forminfo.html');
?>
```

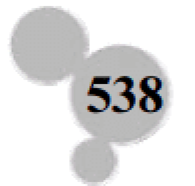
模板页 forminfo.html 包含了两个表格，第一个表格显示下单人所填信息，第二个表格为购买的商品信息。该页面最后一行还特别提醒用户不要忘记订单号，便于日后的订单查询。如果有条件，还可以将订单打印出来。模板页代码如下：

【例 24.50】 代码位置：光盘\TM\sl\24\forminfo.html

```
<!--   下单人信息   -->
<table border="0" align="center" cellpadding="0" cellspacing="0">
  <tr><tdcolspan="4">订单查看 </td></tr>
  <tr><td>订单号：</td><td><font color="red">{$formarr.formid}</font></td>
      <td>订单时间：</td><td>{$formarr.formtime}</td>
  </tr>
  <tr>
    <td>下单人：</td><td>{$formarr.vendee}</td>
    <td>收货人：</td><td>{$formarr.taker}</td>
  </tr>
     …//省略了部分代码
</table>
<!--   商品信息   -->
<table border="0" align="center" cellpadding="0" cellspacing="0">
  <tr><td colspan="5">订单内容</td></tr>
  <tr><td>商品名</td><td>数量</td><td>价格</td><td >价格折率</td><td>合计</td></tr>
{foreach key=key item=item from=$commname}
<tr>
    <td>{$item}</td><td>{$commnumber[$key]}</td>
    <td>{$commagoprice[$key]} 元</td><td>{$commfold[$key]} 折</td>
    <td>{$commagoprice[$key]*$commnumber[$key]} 元</td>
</tr>
{/foreach}
<tr><td colspan="5">总消费：{$formarr.total} 元</td></tr></table>
<p  align="center">恭喜您！ 订单提交成功。<br  />请您在一周内按您的支付方式进行汇款,汇款时注明您的<font color="red">订单号</font>!汇款后请及时通知我们</p>
<p ><font color="red">注意：</font>请记住<font color="red">订单号</font>。以便日后查询及有疑问时使用。</p>
<p align="center"><input type="button" value="我要打印" onclick="window.print()" class="btn" /> </p>
```

24.12.5 查询订单

订单提交后，可以通过导航栏中的“订单查询”超链接来查看订单情况。查询关键字可以是会员



名称或订单号，只要输入一项即可。单击“查询”按钮将会在页面下方显示查询结果，查询订单页面的运行结果如图 24.28 所示。

查询订单			
查询用户：	mrsoft	查询订单号：	
查询			

订单号	订货会员	收货人	订单金额	付款方式	收款方式	订单状态
1259913994	mrsoft	潘攀	9019	邮局汇款	送货上门	未作处理
1259998801	mrsoft	mrsoft	8764	银行转账	平邮	未作处理

图 24.28　查询订单页面的运行结果

本模块采用了 Ajax 技术，将查询结果在本页直接显示出来。一共使用了 4 个文件：queryform.php、queryform.html、queryform.js 和 query.php。

queryform.php 页很简单，载入 Smarty 配置文件后，直接使用 display 方法显示 queryform.html 即可。

queryform.html 包含了一个查询表单和一个 div 标签，查询表单元素和 div 标签属性及说明如表 24.3 所示。

表 24.3　查询表单元素和 div 标签属性及说明

元 素 名 称	元 素 属 性	元 素 说 明
<form>	id='queryform' name='queryform' method='post' action='#'	查询表单，名称为 queryform
text	input id='name' name='name' type='text' class='txt'	会员查询文本框
text	id='formid' name='formid' type='text' class='txt'	订单号查询文本框
button	id='enter' name='enter' type='button' value='查询' class='btn' onClick='return queryrst(queryform)'	“查询”按钮，调用 queryrst()函数
<div>	id=exam	用于显示查询结果

通过表 24.3 可以看到，当单击“查询”按钮时将会触发 onclick 事件，调用 queryrst()函数。queryrst()函数在脚本文件 js/queryform.js 内。该文件的代码如下：

【例 24.51】 代码位置：光盘\TM\sl\24\js\queryform.js

```
function queryrst(form) {
	/*	获取会员名称	*/
	var name = form.name.value;
	/*	获取订单号	*/
	var formid = form.formid.value;
	/*	如果两者都为空，则弹出警告框	*/
	if(name == '' && formid == ''){
		alert('用户或订单号至少有一个不能为空');
		form.name.focus();
		return false;
	}
	/*	将获取的变量传给 query.php 页	*/
	var url = "query.php?vendee="+name+"&formid="+formid;
	xmlhttp.open("GET",url,true);
	/*	调用 showfm()函数	*/
	xmlhttp.onreadystatechange = showfm;
```

```
        xmlhttp.send(null);
}
function showfm(){
        if(xmlhttp.readyState == 4){
                var msg = xmlhttp.responseText;
                if(msg == '0'){
        /*      如果没有查询结果，则清空 exam      */
                                exam.innerHTML = '';
                                alert('没有查询结果');
                }else{
        /*      将结果输出到页面      */
                                exam.innerHTML = msg;
                }
        }
}
```

query.php 页根据传递的 vendee 和 formid 参数，从 tb_form 表中查询记录，如果没有记录，则返回 0；如果有记录，则输出完成的表单字串。query.php 页的代码如下：

【例 24.52】 代码位置：光盘\TM\sl\24\query.php

```
<?php
        /*      载入数据库连接文件      */
        include_once 'inc/char.php';
        include_once 'system/system.inc.php';
        /*      获取 url 参数      */
        $vendee = $_GET['vendee'];
        $formid = $_GET['formid'];
        $reback = '0';
        /*      创建一个表格，并将表单存储到临时变量中      */
        $tmp="<table width='630' border='0' align='center' cellpadding='0' cellspacing='0'><tr><td height='24' colspan
='7' align='center' valign='middle' class='first'>查询结果</td></tr><tr><td width='100' height='24' align='center'
valign='middle' class='left'>订单号</td><td width='75' height='24' align='center' valign='middle' class='center'>订货会
员</td><td width='75' height='24' align='center' valign='middle' class='center'>收货人</td><td width='100' height='24'
align='center' valign='middle' class='center'>订单金额</td><td width='75' height='24' align='center' valign='middle'
class='center'>付款方式</td><td width='75' height='24' align='center' valign='middle' class='center'>收款方式
</td><td width='100' height='24' align='center' valign='middle' class='right'>订单状态</td></tr>";
        /*      根据参数值，生成查询语句      */
        $sql = "select id,formid,vendee,taker,total,pay_method,del_method,state from tb_form where vendee = '".
$vendee."' or formid = '".$formid."'";
        /*      执行 sql 语句，返回查询结果集      */
        $rst = $conn->execute($sql);
        if($rst->RecordCount() == 0){
        }else{
        /*      如果有查询记录，则将结果循环输入到表格内      */
                $arr = $rst->getArray();
                foreach($arr as $value){
                        $tmp.= "<tr><td height=24 align=center valign=middle class=left>".$value['formid']."</td><td align=
center valign=middle class=center>".$value['vendee']."</td><td align=center valign=middle class= center>". $value
['taker']."</td><td align=center valign=middle class=center>".$value['total']."</td><td align=center valign= middle class
=center>".$value['pay_method']."</td><td align=center valign=middle class=center>".$value['del_method']."</td>
<td align=center valign=middle class=right>";
```

```
    /*      对订单状态进行处理          */
                switch (state){
                    case 0:$tmp .= '未作处理';break;
                    case 1:$tmp .= '已付款';break;
                    case 2:$tmp .= '已发货';break;
                    case 3:$tmp .= '已收货';break;
                }
                $tmp .= "</td></tr>";
            }
            $tmp .= "</table>";
    /*      将临时变量赋给$reback 并输出                  */
            $reback = $tmp;
    }
    echo $reback;
?>
```

24.13　后台首页设计

视频讲解：光盘\TM\lx\24\后台首页设计.exe

本模块使用的数据表：tb_form

后台管理系统是网站管理员对商品、会员及公告等信息进行统一管理的场所，本系统的后台主要包括以下功能。

- ☑ 类别管理模块：主要包括对商品类别的添加、修改及删除操作。
- ☑ 商品管理模块：主要包括对商品的添加、修改、删除及订单处理。
- ☑ 用户管理模块：主要包括管理员管理和会员管理。其中管理员管理是实现对管理员的添加、删除和修改功能，会员管理则包括删除和冻结功能。
- ☑ 公告管理模块：主要包括公告的添加及删除操作。
- ☑ 链接管理模块：主要包括添加、修改和删除友情链接。

后台首页的运行结果如图 24.29 所示。



图 24.29　后台首页的运行结果

24.13.1 后台首页布局

后台首页和前台首页不同，其使用的是框架布局。main.html 页的框架代码如下：

【例 24.53】 代码位置：光盘\TM\sl\24\main.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<link rel="stylesheet" href="css/style.css" />
<title>鹏逊电子商务后台管理系统</title>
</head>
<!--  使用 frame 框架布局  -->
<frameset rows="126,*" cols="*" frameborder="no" border="0" framespacing="0">
<!--  头部框架，命名为 topFrame，载入文件为 top.html  -->
  <frame src="top.php" name="topFrame" scrolling="No" noresize="noresize" id="topFrame" title="topFrame" />
  <frameset rows="*" cols="210,*" framespacing="0" frameborder="no" border="0">
<!--  左侧框架，名字为 leftFrame，载入文件为 leff.php  -->
    <frame src="left.php" name="leftFrame" frameborder="0" scrolling="auto" noresize="noresize" id="leftFrame"
title= "leftFrame" />
<1--  中部框架，名字为 mainFrame，载入文件为 default.php  -->
    <frame src="default.php" name="mainFrame" id="mainFrame" title="mainFrame" />
  </frameset>
</frameset>
<noframes><body></body></noframes>
</html>
```

24.13.2 DIV+JavaScript+CSS 实现树形菜单

该首页包含了 3 个文件：top.html、left.php 和 default.php。这里需要注意的是，left.php 页是一个树形菜单，是应用 DIV+JavaScript+CSS 来实现的。首先来介绍一下 div 标签。

【例 24.54】 代码位置：光盘\TM\sl\24\admin\left.html

```
<!--  载入 css 样式和 javascript 脚本  -->
<link href="css/left.css" rel="stylesheet" type="text/css" />
<script language="javascript" src="js/left.js"></script>
<!--  类别管理菜单，注意加粗的地方  -->
<div id="type" align="center" onclick="javascript:change(one,type);">类别管理</div>
<!--  子菜单  -->
<div id="one" style="display: ">
<div id="addtype" align="center"><a href="addtype.php" target="mainFrame" id="menu">添加类别</a></div>
<div id="showtype" align="center"><a href="showtype.php" target="mainFrame" id="menu">查看类别</a></div>
</div>
<div id="hidediv" align="center"></div>
<!--  商品管理菜单  -->
```



```
<div id="commo" align="center" onclick="javascript:change(two,type);">类别管理</div>
<div id="two"style="display:none">
<!--   商品管理子菜单   -->
      …
</div>
…
```



说明

除了加粗的 id 名称和 js 事件不同外，其他菜单的结构完全相同，此时只需修改超链接即可。

注意

除了第一个类别菜单的子菜单 display 样式为空外，其他几个子菜单的 div 样式都为 display=none;。

该页面在 Dreamweaver 中的效果如图 24.30 所示。

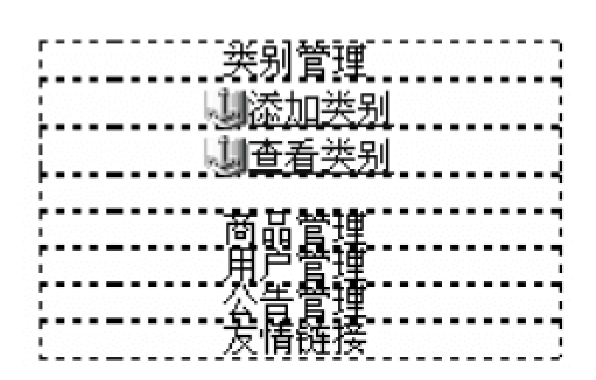


图 24.30　div 树形菜单

因为其他子菜单的样式为 display=none，所以只有“类别管理”子菜单是可见的，下面为它添加 JavaScript 事件。left.js 脚本文件的代码如下：

【例 24.55】 代码位置：光盘\TM\sl\24\admin\js\left.js

```
/*  函数功能：改变子菜单样式，更换当前菜单的背景图片
 * nu：为子菜单 div 标签的 id 号
 * lx：为调用 change()函数所在的标签 id
 */
function change(nu,lx){
/*     如果当前子菜单的样式为 none，则显示子菜单 */
       if(nu.style.display == "none"){
              nu.style.display = "";
              lx.style.background="url(images/main_openroot.gif)";
/      否则，隐藏当前菜单的子菜单   */
       }else{
              nu.style.display = "none";
              lx.style.background="url(images/main_closeroot.gif)";
}}
```

从上述实例可以看出，很简单的函数却实现了很灵活的功能。最后在 left.css 中设置 div 的长和宽等一些默认参数。一个简单而又实用的树形菜单就完成了。

对于后台的大部分模块来说，其功能实现的方法和开发步骤在前台的那些模块设计中基本都已经介绍过。这里只对类别管理模块和商品管理模块中的订单管理进行讲解。

24.14 类别管理模块设计

视频讲解：光盘\TM\lx\24\类别管理模块设计.exe

类别管理模块的功能是方便管理员添加、修改及删除商品类别，本系统只提供二级类别的管理和维护。类别管理模块的实现难点在于如何动态地存取类别记录，使一级类别和二级类别自动关联。首先来看添加类别的实现过程。

24.14.1 添加类别

添加类别的操作流程是：当用户单击“添加类别”超链接时，将在右侧主框架区显示“添加商品类别”表单，用户可以在“类别名称”文本框中输入类别名，还可以通过“类别等级”下拉列表框选择添加商品的类别。当选择“一级类别”时，下面的“父类名称”则不可用。添加类别页面的运行结果如图 24.31 所示。



图 24.31　添加类别页面的运行结果

首先来介绍 addtype.php 页。当单击“添加类别”超链接时，就是调用的该 PHP 页。该页从数据库中读取所有 supid=0，也就是一级类别的记录，并将结果集直接输出为下拉菜单，传递给模板页。addtype.php 页的代码如下：

【例 24.56】 代码位置：光盘\TM\sl\24\admin\addtype.php

```
<?php
    /*    载入数据库连接文件和 Smarty 模板配置文件    */
    include_once 'system/system.inc.php';
    /*    生成查询语句，查询所有 supid=0 的一级类别  */
    $sql = "select name,id from tb_class where supid = 0";
    /*    执行 sql 语句，返回结果集       */
```



```
	$rst = $conn->execute($sql);
	/*	将结果集直接输出为下拉菜单，并传递给模板	*/
	$smarty->assign('op',$rst->GetMenu2("supid",'',$blank = false,",",'class="txt"'));
	$smarty->display('addtype.html');
?>
```

模板页 addtype.html 接收 op 下拉菜单后，将其显示在“添加商品类别”表单中。addtype.html 模板的代码如下：

【例 24.57】 代码位置：光盘\TM\sl\24\admin\addtype.html

```
<!--	载入 js 脚本文件	-->
<script language="javascript" src="js/createxmlhttp.js"></script>
<script language="javascript" src="js/chktype.js"></script>
<table width="300" border="0" align="center" cellpadding="0" cellspacing="0">
<form id="addtype" name="addtype" method="post" action="#">
	<tr><td colspan="2">添加商品类别</td></tr>
	<tr><td >类别名称：</td><td><input name="names" type="text" id="names" class="txt"></td></tr>
	<tr><td >类别等级：</td><td>
	<!--	类别等级下拉菜单，会触发 onchange 事件，调用 changetype()函数	-->
			<select name="grade" OnChange="changetype(addtype)" >
				<option value="1">一级类别</option>
				<option value="2" selected>二级类别</option>
			</select>
	</td></tr>
	<!--	显示 op 变量	-->
	<tr><td>父级名称：</td> <td>{$op}</td></tr>
	<tr><td colspan="2"><input type="button" value="添加" class="btn" onClick="chktype(addtype)"></td>
	</tr>
</form>
</table>
```

该模板通过“类别等级”下拉菜单的 changetype()函数，来控制 op 菜单的状态。如果添加的是一级类别，则 op 菜单不可用；如果添加的是二级菜单，则 op 菜单可用。当用户输入添加类别后，单击“添加”按钮时，会调用 chktype()函数。包含这两个函数的 chktype.js 脚本文件的代码如下：

【例 24.58】 代码位置：光盘\TM\sl\24\admin\js\chktype.js

```
/*	控制 op 菜单的状态	*/
function changetype(form){
		if(form.grade.value=="2"){
			form.supid.disabled=false;
		}else{
			form.supid.disabled=true;
}}
/*	处理类别名称文本框	*/
function chktype(form){
	var name = form.names.value;
	var supid = form.supid.value;
	if(form.names.value == ""){
```

```
            alert('请填写类别名称');
            form.names.focus();
            return false;
        }
/*      根据不同的类别等级，生成不同的 url      */
        if(form.grade.value == "1"){
            var url = "chktype.php?name="+name;
        }else{
            var url = "chktype.php?name="+name+"&supid="+supid;
        }
/*      使用 xmlhttp 对象打开该 url，并调用 check()函数处理返回值  */
        xmlhttp.open("GET",url,true);
        xmlhttp.onreadystatechange = check;
        xmlhttp.send(null);
}
/*      根据返回值作出相应的处理      */
function check(){
        if(xmlhttp.readyState == 4){
            if(xmlhttp.status == 200){
                var msg = xmlhttp.responseText;
                if(msg == "1"){
                    alert('类名重复');
                }else if(msg == "2"){
                    alert('添加失败!');
                }else{
                    alert('添加成功');
                    location='showtype.php';
}}}}
```

在 chktype.php 处理页中实现了对类别的添加，并返回不同的参数以便 chktype()函数处理。chktype.php 页的代码如下：

【例 24.59】 代码位置：光盘\TM\sl\24\admin\chktype.php

```
<?php
    include_once 'system/system.inc.php';                              //载入数据库连接文件
    $name = $_GET['name'];                                             //获取类别名称
    ($_GET['supid'] == '')?($supid = 0):($supid = $_GET['supid']);     //获取类别等级
    $reback = '';                                                      //设置返回变量
    $sql = "select * from tb_class where name = '$name'";              //声明 sql 语句
    $rst = $conn->execute($sql);                                       //返回查询结果集
    if($rst->RecordCount() == 1){                                      //如果有查询结果，说明类别名被占用
        $reback = '1';
    }else{                                                             //否则，执行添加操作
        $ine = array();
        $ine["name"] = $name;
        $ine["supid"] = $supid;
        $intsql = $conn->GetInsertSQL($rst,$ine);                      //生成添加语句
        if($conn->execute($intsql) == false){
            $reback = '2';
```

```
        }else{
            $reback = '3';                              //如果成功，返回 3
        }}
    echo $reback;
?>
```

24.14.2　查看类别

类别添加成功后，可以通过“查看类别”超链接来修改或删除类别。在查看商品页中，类别按照父类 1：子类 1、子类 2……父类 2：子类 1、子类 2……的格式显示。修改类别名称时，直接在类别文本框中输入新类名，然后单击对应的“修改”按钮即可，删除时则直接单击对应的“删除”按钮。当要删除的父类含有子类时，则不允许删除父类。查看类别页面的运行结果如图 24.32 所示。

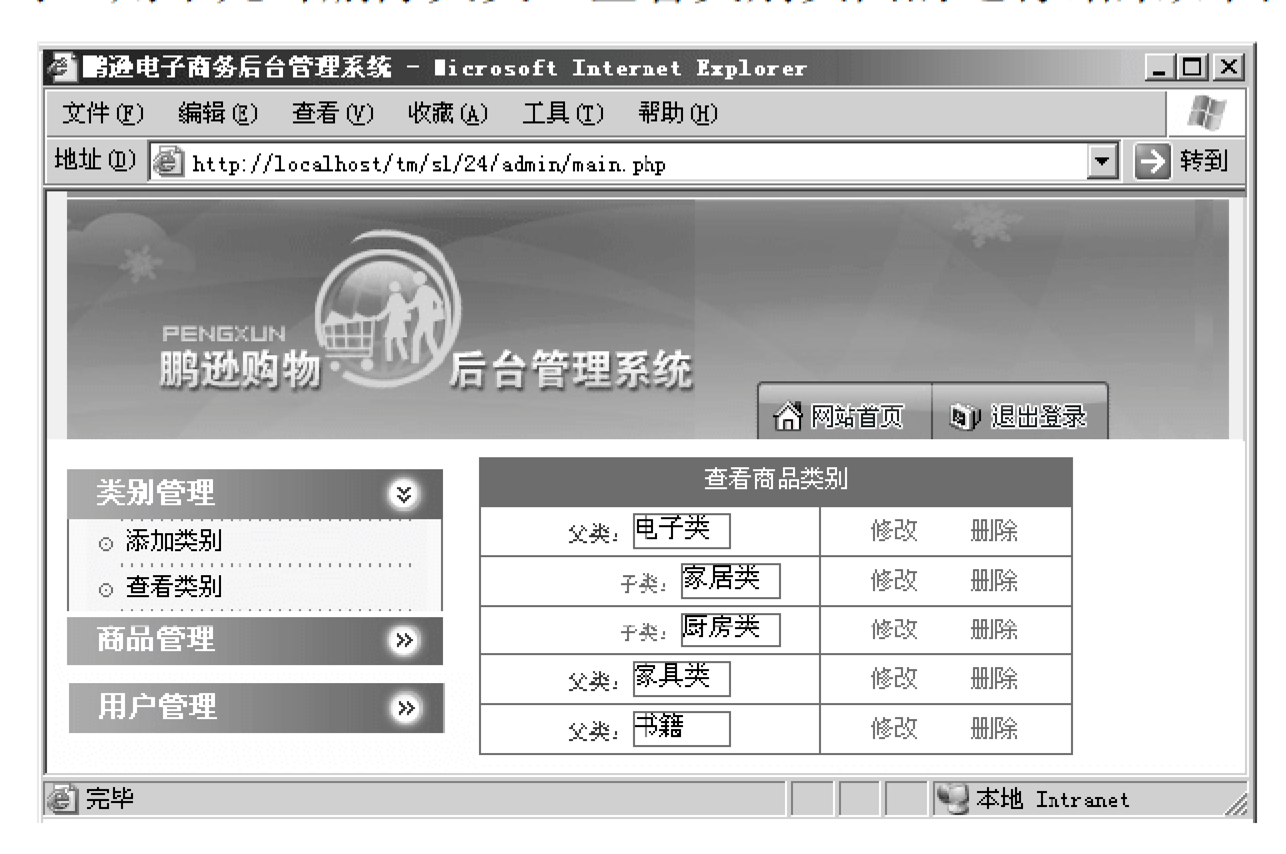


图 24.32　查看类别页面的运行结果

首先来介绍 showtype.php 页。该页执行两次查询，分别输出一级类别和二级类别的数组，传递给模板页。showtype.php 页的代码如下：

【例 24.60】 代码位置：光盘\TM\sl\24\admin\showtype.php

```
<?php
    include_once 'system/system.inc.php';;                  //载入 Smarty 配置文件和数据库连接文件
    $bigsql = 'select id,name from tb_class where supid = 0';  //生成一级类别查询语句
    $smallsql = 'select * from tb_class where supid != 0';     //生成二级类别查询语句
    $bigclass = $conn->execute($bigsql);                    //返回一级类别结果集
    $smallclass = $conn->execute($smallsql);                //返回二级类别结果集
    $bigarray = $bigclass->GetAssoc();                      //将结果集输出成数组类型
    $smallarray = $smallclass->GetAssoc();
    $smarty->assign('bigarray',$bigarray);                  //将数组传递给模板
    $smarty->assign('smallarray',$smallarray);
    $smarty->display('showtype.html');
?>
```

showtype.html 模板页通过嵌套 foreach 循环，将一级类别和二级类别一一对应起来，并输出到页面。

showtype.html 页的代码如下：

【例 24.61】 代码位置：光盘\TM\sl\24\admin\showtype.html

```
<!--   载入 js 脚本文件-->
<script language="javascript" src="js/createxmlhttp.js"></script>
<script language="javascript" src="js/changetype.js"></script>
<table>
<!--   省略 html 无用代码   -->
…
<!--   外层循环，输出一级类别   -->
{foreach name=ftype key=fkey item=fitem from=$bigarray}
     <tr><td>父类：
     <input id="moditype{$fkey}" name="moditype{$fkey}" type="text" class="shorttxt" value="{$fitem}" style="border-color:#996633;" /></td>
         <td>
<input id = " modify " name = " modify " type = " button " value = "  修改  " onclick = " modifytype({$fkey}); "/>
<input id = " delete " name = " delete " type = " button " value = "删除" onclick = "delbigtype({$fkey});">
      </td></tr>
<!--   内层循环，输出对应的二级类别   -->
     {foreach name = stype key = skey item = sitem from = $smallarray}
     {if $sitem[1] == $fkey}
     <tr><td>子类：
     <input id="modtype{$skey}" name="moditype{$skey}" type="text" value="{$sitem[0]}"/></td>
          <td><input id="modidfy" name="modify" type="button" value="修改" onclick=" modifytype({$skey})"/>
          <input id="delete" name="delete" type="button" value="删除"onclick=" delsmalltype({$skey})">
          </td></tr>
      {/if}
     {/foreach}
{/foreach}
</form>
</table>
```

showtype.html 模板中包含的 changetype.js 脚本文件主要有 3 个函数：modifytype()、delbigtype()和 delsmalltype()，分别控制类别的修改和删除。该脚本文件的代码如下：

【例 24.62】 代码位置：光盘\TM\sl\24\admin\js\changetype.js

```
/*    修改类别名称函数    */
function modifytype(key){
      var nm = 'moditype'+key;
      var names = document.getElementById(nm).value;                    //获取类别名称
      if(names == ""){
            alert('请填写类别名称');
            document.getElementById(nm).focus();
            return false;
      }
/*    将类名、id 传给 changetype.php 页      */
      var url = "changetype.php?action=m&names="+names+"&key="+key;
      xmlhttp.open("GET",url,true);
/*    调用 check()函数，返回处理结果     */
```

```
        xmlhttp.onreadystatechange = check;
        xmlhttp.send(null);
}
/*      删除大类，解释同上          */
function delbigtype(key){
        if(confirm("您要删除的是一级类别，您确定要删除吗？")){
                var url = "changetype.php?action=bd&key="+key;
                xmlhttp.open("GET",url,true);
                xmlhttp.onreadystatechange = check;
                xmlhttp.send(null);
        }else{
                return false;
}}
/*      删除小类，解释同上          */
function delsmalltype(key){
        if(confirm("确定要删除选中的项目吗？一旦删除将不能恢复！")){
                var url = "changetype.php?action=sd&key="+key;
                xmlhttp.open("GET",url,true);
                xmlhttp.onreadystatechange = check;
                xmlhttp.send(null);
        }else{
                return false;
}}
/*      根据不同的返回结果做不同的操作      */
function check(){
        if(xmlhttp.readyState == 4){
                if(xmlhttp.status == 200){
                        var msg = xmlhttp.responseText;
                        if(msg == "1"){
                                alert('类名重复');
                        }else if(msg == "2"){
                                alert('操作失败!');
                        }else if(msg == '3'){
                                alert('操作成功');
                                location='showtype.php';
                        }else if(msg == '4'){
                                alert('该大类有子类，不能删除');
                        }else if(msg == '0'){
                                alert('未知错误!'+'\n 错误代码:'+msg);
}}}}
```

无论是修改还是删除，都是调用 changetype.php 页进行处理的，当进行修改操作时，传递的 action 参数为 m；当删除大类时，action 为 bd；当删除小类时，action=sd。changetype.php 页的代码如下：

【例 24.63】 代码位置：光盘\TM\sl\24\admin\changetype.php

```
<?php
        /**********************************
        $reback 说明：
        0、未知错误 1、类名重复 2、操作失败 3、操作成功 4、有二级分类
```

```
**********************************/
include_once 'system/system.inc.php';
$action = $_GET['action'];                                          //获取动作类型
$reback = '';
/*      修改操作  */
if($action == 'm'){
        /*      获取类别名称和 id    */
        $names = $_GET['names'];
        $key = $_GET['key'];
        /*      检测添加名称是否存在      */
        $sql = "select * from tb_class where name = '$names'";
        $rst = $conn->execute($sql);
        if($rst->RecordCount() == 1){
                $reback = '1';
        }else{
        /*      如果不存在，创建更新语句，执行更新操作      */
                $updatesql = "select * from tb_class where id = ".$key;
                $updaterst = $conn->execute($updatesql);
                $upd = array();
                $upd["id"] = $key;
                $upd["name"] = $names;
                $update = $conn->GetUpdateSQL($updaterst,$upd);        //创建更新语句
                if($conn->execute($update) == false){                  //执行更新操作
                        $reback = '2';
                }else{
                        $reback = '3';
                }
        }
/*      删除小类，根据类别 id，直接删除            */
}else if($action == 'sd'){
        $key = $_GET['key'];                                           //类别 id
        $delsql = "delete from tb_class where id = ".$key;             //删除语句
        if($conn->execute($delsql) == false){                          //执行删除操作
                $reback = '2';
        }else{
                $reback = '3';
        }
/*      删除大类          */
}else if($action == 'bd'){
        $key = $_GET['key'];
        $sql = "select * from tb_class where supid = ".$key;
        $rst = $conn->execute($sql);
        /*      如果要删除的大类有子类存在，则不做删除处理       */
        if($rst->RecordCount() >= 1){
                $reback = '4';
        }else{
        /*      否则，删除该小类    */
                $delsql = "delete from tb_class where id = ".$key;
                if($conn->execute($delsql) == false){
```

```
                    $reback = '2';
                }else{
                    $reback = '3';
                }
            }
    }else{
        $reback = '0';
    }
    echo $reback;
?>
```

24.15　订单管理模块设计

视频讲解：光盘\TM\lx\24\订单管理模块设计.exe

订单管理模块的功能包括查看、删除和处理订单等操作。其中，查看订单功能和 24.12.4 节的反馈订单实现方法和步骤类似，而删除操作和 24.11.4 节的删除商品模块类似，所以，这里只给出处理订单的功能实现。处理订单是指将用户提交的订单进行处理。默认的订单状态是“未处理”，可以将订单的状态修改为“已收款”、“已发货”和“已收货”3 种。处理订单页面的运行结果如图 24.33 所示。



图 24.33　处理订单页面的运行结果

限于篇幅，这里只给出与订单相关的程序代码。首先来介绍模板页 showform.html，该页显示所有订单的记录，在每条记录的后面可以对订单进行查看及处理。当单击“请求处理”按钮时，该页面下方的隐藏表单将变为显示状态，并将订单号显示在该表单中，该隐藏表单包含一组单选按钮，可以对表单状态进行修改。showform.html 页的代码如下：

【例 24.64】 代码位置：光盘\TM\sl\24\admin\showform.html

```
<!--   载入 js 脚本文件   -->
<script language="javascript" src="js/createxmlhttp.js"></script>
<script language="javascript" src="js/shwform.js"></script>
```

```
<table border="0" cellspacing="0" cellpadding="0" align="center">
  <form id="shwfrm" name="shwfrm" method="post" action="#">
    <tr>
<!--  省略部分为要显示的订单信息的字段名   -->
     …
      <td>订单状态</td>
      <td>处理</td>
    </tr>
    {foreach key=key item=item from=$formarr}
    <tr>
<!--  省略部分为显示的订单信息   -->
     …
      <td>
<!--  根据不同的状态值，显示不同的结果   -->
{if $item.state == 0}未处理{elseif $item.state == 1}已收款{elseif $item.state == 2}已发货{elseif $item.state == 3}
已收货{/if}
      </td>
<!--  单击“请求处理”按钮，调用 showme()函数   -->
      <td><input id="deal" name="deal" type="button" value="请求处理" onclick="showme({$item.formid})" />
</td>
    </tr>
    {/foreach}
    …
  </form>
</table>
<!--  隐藏的表单，id 为 chdl   -->
<div id="chdl" style="display:none;">
  <table width="524" border="0" align="center" cellpadding="0" cellspacing="0">
    <form id="changedeal" name="changedeal" method="post">
      <tr><tdcolspan="6"">订单处理</td></tr>
      <tr><td>订单号：</td>
<!--  当隐藏表单显示时，该<div>标签将显示被激活订单的 formid   -->
        <td><div id="formid"> </div></td>
<!--  下面为一组单选按钮，它们有着相同的名称和不同的 id   */
        <td><input id="acceptmon" name="acceptsend" type="radio" value="1" checked="checked" />已收款
</td>
        <td><input id="sendwa" name="acceptsend" type="radio" value="2" />已发货</td>
        <td><input id="acceptwa" name="acceptsend" type="radio" value="3" />已收货</td>
<!--  单击“修改”按钮，会调用 changeme()函数来更新订单状态   -->
        <td><input id="chg" name="chg" type="button" value="修改" onclick="return changeme(changedeal)"
/></td>
      </tr>
    </form>
  </table>
</div>
```

在该页中所调用的函数都包含在 showform.js 脚本文件中。showme()函数的作用是改变 id=chdl 的<div>标签的状态，唯一的参数是该条订单的订单号（formid），并在 id=formid 的<div>标签中显示出来；另一个函数 changeme()的作用是将 formid 值和被选中的单选按钮值取出并传给 changestate.php 进

行处理，根据处理返回值刷新本页面。showform.js 页面的主要代码如下：

【例 24.65】 代码位置：光盘\TM\sl\24\admin\js\showform.js

```
/*    显示/隐藏表单  */
function showme(formid){
      /*    判断“id=chdl”的<div>标签的样式，如果是显示，则改为隐藏     */
      if(document.getElementById('chdl').style.display == ''){
            document.getElementById('chdl').style.display = 'none';
      /*    否则，将该标签显示出来，并将 formid 显示在“id=formid”的<div>标签中   */
      }else{
            document.getElementById('chdl').style.display = '';
            document.getElementById('formid').innerHTML = formid;
      }
}
/*    改变状态  */
function changeme(form){
      /*    获取 formid     */
      var fid = document.getElementById('formid').innerText;
      var lang = form.acceptsend.length;
      var state;
      /*    循环查看单选按钮的状态，并取得被选中按钮的 value 值  */
      for(var i=0; i < lang; i++){
            if(form.acceptsend[i].checked == true){
                  state = form.acceptsend[i].value;
            }
      }
      if(state == undefined){
            alert('请选择处理项');
            return false;
      }
      /*    将 formid 和 state 值传给 changestate.php 页*/
      var url = "changestate.php?formid="+fid+"&state="+state;
      /*    使用 xmlhttpprequest 对象调用该 url        */
      xmlhttp.open("GET",url,true);
      /*    对返回进行处理响应        */
      xmlhttp.onreadystatechange = function(){
            if(xmlhttp.readyState == 4){
                  var msg = xmlhttp.responseText;
                  if(msg == '1'){
                        location.reload();
                  }else{
                        alert('修改失败'+msg);
}}}
      xmlhttp.send(null);
}
```

最后的 changestate.php 负责将改变的记录更新到数据表中。changestate.php 页的代码如下：

【例 24.66】 代码位置：光盘\TM\sl\24\admin\changestate.php

```
<?php
    include_once 'system/system.inc.php';;                          //调用数据库连接文件
    $formid = $_GET['formid'];                                      //获取参数值
    $state = $_GET['state'];
    $reback = '0';                                                  //声明返回值，初始化为 0
    $arr = array();                                                 //声明更新数组
    $sql = "select * from tb_form where formid = '".$formid."'";    //根据 formid 生成查询语句
    $rst = $conn->execute($sql);                                    //返回查询结果
    if($rst->RecordCount() != 1){                                   //如果没有返回值，返回 2
        $reback = '2';
    }else{                                                          //如果有返回值
        $arr['state'] = $state;                                     //将更新数据存到数组中
        $UpdateSQL = $conn->GetUpdateSQL($rst, $arr);               //生成更新语句
        if(false == $conn->execute($UpdateSQL)){                    //执行更新
            $reback = '3';
        }else{
            $reback = '1';                                          //成功返回 1
}}
    echo $reback;                                                   //输出返回值
?>
```

24.16　开发常见问题与解决

视频讲解：光盘\TM\lx\24\开发常见问题与解决.exe

在本系统开发和后期测试的过程中，开发人员遇到了各种各样的疑难问题。这里找出一些常见的、容易被忽略的问题加以讲解，希望能够为初学者和新手提供一些帮助，在开发程序时少走一些弯路。

24.16.1　解决 Ajax 的乱码问题

问题描述：当使用 Ajax 传递数据时，要么在数据处理页中数据不能被正确处理，要么输出返回值时显示的是一堆无法识别的乱码。

解决方法：这是因为 PHP 在传递数据时使用的编码默认为 UTF-8，这就造成了非英文字符不能正确传递的情况。解决方法如下：

在所有的 PHP 页中都输入代码“Header(“Content-Type:text/html;charset=gb2312”);”这样，所有的页面即可正确显示。

24.16.2　使用 JS 脚本获取、输出标签内容

问题描述：获取、更改表单元素值和特定标签内容。

解决方法：使用 JS 脚本获取页面内容的方式主要有两种，第一种是通过表单获取表单元素的 value



值。格式为：表单名称.元素名.value。该方式只能获取表单中的元素值，对于其他标签元素不适用。而第二种方式可以通过 id 名来获取页面中任意标签的内容。格式为：document.getElementById('id'). value;或 document.getElementById ('id').innerText;。

使用第二种方式时要注意，标签的 id 名必须存在且唯一，否则就会出现错误。为标签内容赋值时，则使用如下格式：

```
id.innerHTML ='要显示的内容';
```

24.16.3　使用浮动框架做关联菜单

问题描述：在添加商品时，商品类别选项是一个关联菜单。如果要在不刷新本页的前提下实现该功能十分复杂（若使用了 ADODB 和 Ajax 会较为简便）。

解决方法：这里使用了一个比较灵活的方法，就是使用浮动框架技术，将一个二级管理菜单放到单独的一个网页中，然后在框架中显示。当选择不同类别时，可以刷新菜单页，但整个表单页却不会同时刷新。其代码如下：

```
…
<tr>
<td>商品类别：</td>
<td>
<iframe id="menu1" name="menu1" src="menu.php" width="300" height="22" frameborder="0" scrolling="no" style=" margin-top:0px; margin-left: 0px; top:0px;"></iframe>
<input id="stype" name="stype" type="hidden" value="" />
</td>
</tr>
…
```

加粗的代码部分为隐藏表单，保存当前被选择的类别 id，添加商品处理页就是从这个隐藏表单中获取商品类别的。

24.16.4　禁用页面缓存

问题描述：使用 Ajax 技术可以防止页面刷新，但有时也会产生新的问题。如在“会员管理”页面，如果连续地“冻结”和“解冻”会员，那么超过 3 次后，该功能将失效，因为在一定时间内，如果做相同的操作，那么 xmlhttprequest 对象会执行缓存中的信息，从而造成操作失败。

解决办法：使用 header()函数将缓存关闭。将代码 header(“CACHE-CONTROL:NO-CACHE”);添加到 xmlhttprequest 对象所调用的处理页的顶部即可。

24.16.5　在新窗口中使用 session

问题描述：使用 js 的 open 方法打开新窗口时，原浏览器中的 session 值不会被传递到新窗口中，

从而造成数据查询失败。

解决方法：将 session 值另存到隐藏域或随着 url 一起传递到新窗口。代码如下：

```
<!--  在模板页中，将 session 值赋给隐藏域   -->
<input id="uid" name="uid" type="hidden" value="{$smarty.session.id}">
…
/*    在 js 脚本中，获取到隐藏域 value 值      */
function getInput(){
      Var uid = document.getElementById('uid').value;
/*    将获取的 value 值通过 url 传给新页面      */
      open("operator.php?uid="+uid,'_blank','',false);
      …
}
```

24.16.6　防止站外链接

问题描述：用户可以在没有登录的情况下访问某些网页，或者通过站外访问、盗链等。

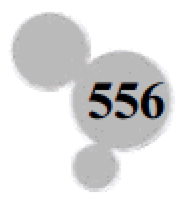
解决方法：通过预定义变量来防止站外链接。$_SERVER['HTTP_REFERER']变量可以获取前一页的 url，如果该变量为空，说明用户是直接从地址栏中输入地址的；如果该变量的服务器地址和系统服务器地址不符，说明是非法用户站外链接。该功能的完整代码如下：

```
<?php
      session_start();                                                          //开启 session 支持
      if(!isset($_SESSION['id']) or !isset($_SESSION['member'])){
            echo "<script>alert('您没有登录或超时');history.back;</script>";       //验证 session
            exit();
      }
      $ref = $_SERVER['HTTP_REFERER'];
      if($ref == ''){
      echo '对不起，不允许从地址栏访问!';
      }else{
            $url = parse_url($ref);
            if($url[host] != '124.0.0.1' && $url[host] != 'localhost'){
                  echo '不允许盗链';
                  exit();
      }}
?>
```

24.16.7　判断上传文件格式

问题描述：添加商品时可以上传商品的图片，但有时可能会误传非图片格式的文件，这里就自定义一个函数来判断上传文件的后缀。

解决方法：创建自定义函数 f_postfix()，函数的代码如下：



```
/*
 *判断文件后缀
 *$f_type：允许文件的后缀类型（数组）
 *$f_upfiles：上传文件名
 */
function f_postfix($f_type,$f_upfiles){
	$is_pass = false;
	$tmp_upfiles = split("\.",$f_upfiles);                    //使用 split()函数分隔文件
	$tmp_num = count($tmp_upfiles);                           //查找文件后缀
	if(in_array(strtolower($tmp_upfiles[$tmp_num - 1]),$f_type))   //判断后缀是否在允许列表内
		$is_pass = $tmp_upfiles[$tmp_num - 1];                //如果是，则将后缀名赋给变量
	return $is_pass;                                          //返回变量
}
```

24.16.8　打开 Smarty 缓存文件

在开发系统时，要将 Smarty 的缓存关闭，避免出现一些莫名其妙的错误。当系统开发完毕后，就要将缓存打开，可以大大提高页面访问速度。在 Smarty 模板的配置文件 config.php 中将缓存开启，代码如下：

```
$smarty->caching = true;
```

24.17　发 布 网 站

视频讲解：光盘\TM\lx\24\发布网站.exe

电子商务网站开发完成后即可发布网站。要发布网站，需要经过注册域名、申请空间、将域名解析到服务器和上传网站 4 个步骤。首先来介绍注册域名。

24.17.1　注册域名

域名就是用来代替 IP 地址，以方便记忆及访问网站的名称，如 www.163.com 就是网易的域名；www.yahoo.com.cn 就是中文雅虎的域名。域名需要到指定的网站中注册购买，名气较大的有 www.net.com（万网）和 www.xinnet.com（新网）。

购买注册域名的步骤如下：

（1）登录域名服务商网站。

（2）注册会员。如果不是会员则无法购买域名。

（3）进入域名查询页面，查询要注册的域名是否已经被注册。

（4）如果用户欲注册的域名未被注册，则进入域名注册页面并填写相关的个人资料。

（5）填写成功后，单击“购买”按钮。注册成功。

（6）付款后，等待域名开启。

24.17.2　申请空间

域名注册完毕后就需要申请空间了，空间可以使用虚拟主机或租借服务器。目前，许多企业建立网站都采用虚拟主机，这样既节省了购买机器和租用专线的费用，同时也不必聘用专门的管理人员来维护服务器。申请空间的步骤如下：

（1）登录虚拟空间服务商网站。

（2）注册会员（如果已有会员账号，则直接登录即可）。

（3）选择虚拟空间类型（空间支持的语言、数据库、空间大小和流量限制等）。

（4）确定机型后，直接购买。

（5）进入到缴费页面，选择缴费方式。

（6）付费后，空间在 24 小时内开通，随后即可使用此空间。

注意

申请的空间一定要支持相应的开发语言及数据库。例如本系统要求空间支持的语言为 PHP，则数据库可以是 MySQL、MSSQL 等。

24.17.3　将域名解析到服务器

域名和空间购买成功后需要将域名地址指向虚拟服务器的 IP。进入域名管理页面，添加主机记录，一般要先输入主机名，注意不包括域名，如解析 www.bccd.com，只需输入 www 即可，后面的 bccd.com 不需要填写。接下来填写 IP 地址，最后单击“确定”按钮即可。如果想添加多个主机名，则重复上面的操作即可。

24.17.4　上传网站

最后是上传网站程序。上传网站需要使用 FTP 软件，如果使用 Dreamweaver，则可以直接在 Dreamweaver 中上传。这里以 CuteFTP 为例，详细介绍其操作步骤。

（1）打开 FTP 软件。

（2）选择 File/Site-Manager 命令，将弹出站点面板。

（3）单击 New 按钮，新建一个站点。

（4）在 Label for site 中输入站点名。

（5）在 FTP Host Address 中输入域名。

（6）在 FTP site User Name 中输入用户名。



（7）在 FTP site Password 中输入密码。
（8）单击 Edit...按钮，弹出编辑窗口。
（9）取消选中 Use PASV mode 和 Use firewall setting 复选框。
（10）单击“确定”按钮。
（11）单击 Connect 按钮连接到服务器。
（12）连接服务器后，在左侧的本地页面中右击需要上传的文件，选择“上传文件”命令即可。
（13）如果上传过程中出现错误，选择“继续上传”命令即可。
（14）上传成功后，关闭 FTP 软件。

24.18　小　　结

本章使用 Smarty、ADODB、Ajax 等目前的主流技术，实现了一个电子商务平台从系统分析到最后发布的全过程。希望读者能通过这个项目实例，把前面所学到的各种技术消化吸收、融会贯通，并能够学以致用，举一反三。